SIMON BASIC

114 EK PROGRAMLAMA KOMUTLARI

commodore
BİLGİSAYAR

SIMON BASIC
KULLANICI KILAVUZU

C64108

Commodore Italians SPA
Via Fratelli Gracchi 48,
Cinisello Balsamo, Milano, Italy.

Commodore Computer BV
Marksingel, 2e4811 NV Breda, Postbus 720,
4803aS Breda, Netherlands.

Commodore Business Machines Ltd.,
3370, Pharmacy Avenue, Agincourt,
Ontario, M1W 2K4, Canada.

Commodore A.G. Schweiz,
Aeschenvorstadt 57, 4010,
Basel, Switzerland.

Commodore Business Machines Inc.,
1200, Wilson Drive, West Chester,
PA 19380, USA

Commodore Buromaschinen GmbH,
Lyoner Str. 38, Postfach 710126, 6000
Frankfurt, West Germany

Commodore Business Machines Pty. Ltd.,
5, Orion Road, Lane Cove,
New South Wales 2066, Australia.

Commodore Business Machines (UK) Ltd.,
675 Ajax Avenue, Slough Trading Estate,
Slough, Berks. SL1 4BG England.

YAZAR HAKKINDA

David Simon on üç yaşındayken, babası ona doğum günü hediyesi olarak bir COMMODORE bilgisayar hediye etti. O zamandan beri, bilgisayarlara dair yaşının çok ötesinde bir anlayış geliştirdi. SIMON BASIC programı da bu deneyimin ürünü. David, yeni COMMODORE 64'ünün mümkün olduğunca çok komut içermesi arzusuyla motive olmuştu. Diğer mikro ve hatta bazı mini bilgisayarların sunduğu BASIC varyasyonlarını inceledi. Bu listeden, şu anda SIMON BASIC'ı oluşturan 114 komutu bir araya getirdi. COMMODORE, bu on altı yaşındaki öğrencinin çalışmalarını gururla pazarlıyor.

Bu kılavuz, bir kelime işlemci kullanılarak COMMODORE 8000 serisi bilgisayar sisteminde hazırlanmıştır. Dosyalar daha sonra elektronik olarak bir fotodizöre aktarılmış ve dizgi mühendisinin müdahalesi olmadan

THE ELECTRONIC VILLAGE LTD., Londra W4

tarafından dizilmiştir.

Bu kılavuzun hazırlanmasında yardımcı olan Gail Wellington, Steve Beats ve Keith Morris'e özel teşekkürler.

YORUMLAR VE HATA TALEPLERİ

OKUYUCUYA

Bildiğimiz kadarıyla, bu kılavuz baskıya girdiği tarihte teknik ve tipografik olarak doğrudur. Ancak, hataları yakalamak için ne kadar ince eleyip sık dokursak süzgecimizi çalıştırsak da, bazen birkaç hata gözden kaçabiliyor.

Herhangi bir hata fark ederseniz, lütfen bize bildirin. Yorum, eleştiri ve önerileriniz de içtenlikle rica olunur.

Saygılarımla,

Michael G. Smith.

Teknik Yazar

COMMODORE BUSINESS MACHINES (UK), LTD.
675 Ajax Avenue
Trading Estate
Slough, Berkshire SL1 4BG
ENGLAND

TELİF HAKKI - YAZILIM ÜRÜNÜ

Bu yazılım ürünü telif hakkına sahiptir ve tüm hakları saklıdır.

D. S. Software
19 Reddings
Welwyn Garden City
Herts AL8 7LA
U.K.

Bu ürünün dağıtımı ve satışı yalnızca orijinal alıcıya yöneliktir. Bu programların yasal kullanıcıları, yalnızca programları çalıştırmak amacıyla bu programları ortamdan bir bilgisayarın belleğine okuma lisansına sahiptir. Programların güvenlik kopyaları yalnızca kendi kullanımları için alınabilir. Bu ürünü başka herhangi bir amaçla çoğaltmak, kopyalamak, satmak veya başka bir şekilde dağıtmak yasa ihlalidir.

TELİF HAKKI - KILAVUZU

Bu kılavuz telif hakkıyla korunmaktadır ve tüm hakları saklıdır. Bu belge, Yazılım Ürünleri Müdürü COMMODORE BUSINESS MACHINES, LTD.'nin önceden yazılı izni olmaksızın tamamen veya kısmen kopyalanamaz, fotokopisi çekilemez, yeniden basılamaz, tercüme edilemez, herhangi bir elektronik ortama veya makine tarafından okunabilir bir biçime dönüştürülemez veya herhangi bir şekilde çoğaltılamaz.

SORUMLULUK REDDİ

Programlar yayınlanmadan önce COMMODORE tarafından test edilse de, bu yazılımın doğruluğu konusunda herhangi bir iddiada bulunulmamaktadır. COMMODORE ve dağıtımcıları, bu programların kullanımından kaynaklanan herhangi bir kayıp veya hasardan sorumlu tutulamaz. Programlar yalnızca bu anlayış temelinde satılmaktadır. Her bir uygulama, uygulamadan önce kapsamlı bir şekilde test edilmelidir. Kurulum, bakım veya eğitime ihtiyacınız olması durumunda, lütfen COMMODORE bayinize danışın.

İÇİNDEKİLER

BİRİNCİ BÖLÜM-SIMON BASIC TEKNOLOJİSİNE GİRİŞ

1.1 GİRİŞ 1-1
1.2 SIMON BASIC KILAVUZU 1-2
1.3 SIMON BASIC'E BAŞLAMAK 1-4
1.4 SIMON BASIC KOMUTLAR 1-4
1.5 KOMUT GİRİŞİ 1-6
1.6 KURALLAR 1-6

İKİNCİ BÖLÜM-PROGRAMLAMA YARDIMCILARI

2.1 GİRİŞ 2-1
2.2 FONKSİYON TUŞLARINA KOMUT ATAMA 2-2
2.2.1 KEY 2-2
2.2.2 ADDING CARRIAGE RETURNS 2-2
2.2.3 DISPLAY 2-3
2.3 AUTO 2-3
2.4 RENUMBER 2-4
2.5 PAUSE 2-5
2.6 CGOTO 2-6
2.7 RESET 2-6
2.8 MERGE 2-7
2.9 PROGRAM LİSTELEME YARDIMCILARI 2-8
2.9.1 PAGE 2-8
2.9.2 OPTION 2-9
2.9.3 DELAY 2-10
2.10 FIND 2-11
2.11 PROGRAM HATA AYIKLAMA YARDIMCILARI 2-12
2.11.1 TRACE 2-12
2.11.2 RETRACE 2-13
2.12 DUMP 2-14
2.13 COLD 2-15
2.14 PROGRAM GÜVENLİK YARDIMLARI 2-15
2.14.1 GİRİŞ 2-15
2.14.2 DISAPA 2-16
2.14.3 SECURE 2-17
2.15 OLD 2-17

ÜÇÜNCÜ BÖLÜM-GİRİŞ DOĞRULAMASI VE METİN İŞLEME

3.1 GİRİŞ 3-1
3.2 KARAKTER DİZİSİ İŞLEME 3-2
3.2.1 INSERT 3-2
3.2.2 INST 3-3
3.2.3 PLACE 3-4
3.2.4 DUP 3-5
3.2.5 CENTRE 3-5
3.2.6 AT 3-6
3.2.7 USE 3-7
3.3 GİRİŞ DOĞRULAMA KOMUTLARI 3-8
3.3.1 FETCH 3-8
3.3.2 IN KEY 3-9
3.3.3 ON KEY 3-10
3.3.4 DISABLE 3-11
3.3.5 RESUME 3-11

DÖRDÜNCÜ BÖLÜM-EKSTRA SAYISAL YARDIMLAR

4.1 GİRİŞ 4-1
4.2 EK ARİTMETİK OPERATÖRLER 4-1
4.2.1 MOD 4-1
4.2.2 DIV 4-2
4.2.3 FRAC 4-2
4.3 SAYISAL DÖNÜŞÜM 4-3
4.3.1 İKİLİ SAYIDAN ONLU SAYIYA DÖNÜŞTÜRME 4-3
4.3.2 ONALTILI SAYIDAN ONDALIK SAYIYA DÖNÜŞTÜRME 4-3
4.3.3 DÖNÜŞTÜRME KOMUTLARININ BİRLEŞTİRİLMESİ 4-4
4.4 EXOR 4-4

BEŞİNCİ BÖLÜM-DİSKET KOMUTLARI

5.1 GİRİŞ 5-1
5.2 DISK 5-1
5.3 DIR 5-2

ALTINCI BÖLÜM-SIMON BASIC İLE GRAFİKLER

6.1 GİRİŞ 6-1
6.2 EKRAN YAPILANDIRMASI 6-2
6.3 COMMODORE 64 RENKLERİ 6-2
6.4 ÇİZİM TİPLERİ 6-3
6.5 GRAFİK ÇİZİM KOMUTLARI 6-3
6.5.1 COLOUR 6-3
6.5.2 HIRES 6-4
6.5.3 REC 6-5
6.5.4 MULTI 6-5
6.5.5 NRM 6-6
6.5.6 LOW COL 6-6
6.5.7 HI COL 6-7
6.5.8 PLOT 6-8
6.5.9 TEST 6-9
6.5.10 LINE 6-10
6.5.11 CIRCLE 6-10
6.5.12 ARC 6-11
6.5.13 ANGL 6-12
6.5.14 PAINT 6-13
6.5.15 BLOCK 6-14
6.5.16 DRAW 6-14
6.5.17 ROT 6-15
6.5.18 CSET 6-17
6.6 GRAFİK EKRANDA METİN YAZDIRMA 6-18
6.6.1 CHAR 6-18
6.6.2 TEXT 6-19

YEDİNCİ BÖLÜM-EKRAN MANİPÜLASYONU

7.1 GİRİŞ 7-1
7.2 BCKGNDS 7-2
7.3 FLASH 7-3
7.4 OFF 7-4
7.5 BFLASH 7-4
7.6 FCHR 7-5
7.7 FCOL 7-6
7.8 FILL 7-6
7.9 MOVE 7-7
7.10 INV 7-8
7.11 SCROLLING 7-9
7.12 EKRAN VERİLERİNİN DEPOLANMASI VE GERİ ÇAĞIRILMASI 7-10
7.12.1 SCRSV 7-10
7.12.2 SCRLD 7-11
7.13 EKRAN VERİLERİNİ YAZDIRMA 7-11
7.13.1 GİRİŞ 7-11
7.13.2 COPY 7-11
7.13.3 HRDCPY 7-12

SEKİZİNCİ BÖLÜM-YARATIK VE KULLANICI TANIMLI GRAFİKLER

8.1 GİRİŞ 8-1
8.2 YARATIKLAR 8-1
8.2.1 GİRİŞ 8-1
8.2.2 DESIGN 8-2
8.2.3 @ 8-3
8.2.4 CMOB 8-5
8.2.5 MOB SET 8-6
8.2.6 MMOB 8-7
8.2.7 RLOCMOB 8-8
8.2.8 DETECT 8-8
8.2.9 CHECK 8-9
8.2.10 MOB OFF 8-9
8.3 KULLANICI TANIMLI KARAKTERLER OLUŞTURMA 8-10
8.3.1 GİRİŞ 8-10
8.3.2 MEM 8-10
8.3.3 DESIGN 8-12
8.3.4 @ 8-13

DOKUZUNCU BÖLÜM-YAPILANDIRILMIŞ PROGRAMLAMA

9.1 GİRİŞ 9-1
9.2 DURUM TESTİ VE PROGRAM DÖNGÜLERİ 9-1
9.2.1 IF...THEN...ELSE 9-1
9.2.2 REPEAT UNTIL 9-2
9.2.3 RCOMP 9-3
9.2.4 LOOP...EXIT IF...END LOOP 9-4
9.3 PROGRAM PROSEDÜRLERİ 9-5
9.3.1 GİRİŞ 9-5
9.3.2 PROC 9-5
9.3.3 END PROC 9-6
9.3.4 CALL 9-6
9.3.5 EXEC 9-7
9.4 PROGRAM DEĞİŞKENLERİ 9-8
9.4.1 GİRİŞ 9-8
9.4.2 LOCAL 9-8
9.4.3 GLOBAL 9-9

ONUNCU BÖLÜM-HATA TUTMA

10.1 GİRİŞ 10-1
10.2 ON ERROR 10-1
10.3 OUT 10-3
10.4 NO ERROR 10-4

ON BİRİNCİ BÖLÜM-SIMON BASIC İLE MÜZİK YAPMAK

11.1 GİRİŞ 11-1
11.1.1 SES ŞEKİLLENDİRME 11-1
11.1.2 SES DALGALARI 11-2
11.1.3 SES PROGRAMLAMA 11-4
11.2 MÜZİK KOMUTLARI 11-5
11.2.1 VOL 11-5
11.2.2 WAVE 11-5
11.2.3 ENVELOPE 11-8
11.2.4 MUSIC 11-9
11.2.5 PLAY 11-11

ON İKİNCİ BÖLÜM-OKUMA FONKSİYONLARI

12.1 GİRİŞ 12-1
12.2 PENX 12-1
12.3 PENY 12-2
12.4 POT 12-3
12.5 JOY 12-5

ON ÜÇÜNCÜ BÖLÜM-SIMON BASIC PROGRAM ÖRNEKLERİ

13.1 GİRİŞ 13-1
13.2 PROGRAM 1 - ÇOK YÜZLÜ BİR ŞEKİL ÇİZİMİ 13-1
13.3 PROGRAM 2 - KELİME BULMA 13-2
13.4 PROGRAM 3 - HARF KAYDIRICI 13-5
13.5 PROGRAM 4 - ANTİK BİR ARABA 13-8

EK-HATA MESAJLARI

SÖZLÜK

İNDEKS

ŞEKİLLER TABLOSU

ŞEKİL

3-1 TEK BİR 'AT' KOMUTU 3-6
3-2 BİLEŞİK BİR 'AT' KOMUTU 3-7

8-1 MEM ÖNCESİNDE BELLEK YAPILANDIRMASI 8-11
8-2 MEM SONRASI BELLEK YAPILANDIRMASI 8-11

11-1 BİR SES ZARFI 11-2
11-2 ÜÇGEN BİR SES DALGASI 11-2
11-3 DİŞLİ BİR SES DALGASI..... 11-3
11-4 BİR NABIZ/KARE DALGA..... 11-3
11-5 BİR GÜRÜLTÜ DALGASI 11-4

12-1 OYUN ÇUBUĞU DEĞERLERİ 12-5

BİRİNCİ BÖLÜM

SIMON BASIC'E GİRİŞ

1.1 GİRİŞ

SIMON BASIC kartuşu, COMMODORE 64 bilgisayarınızın tüm potansiyelini ortaya çıkarmanızı sağlamak üzere tasarlanmıştır. Bunu, COMMODORE 64'ün standart BASIC'ini tamamlayan 114 ek komut sağlayarak gerçekleştirir. Bu ek komutlar, aşağıda özetlendiği gibi on iki geniş alana ayrılır:

Daha hızlı, daha verimli BASIC programlamayı kolaylaştırmak için KEY ve TRACE gibi PROGRAMLAMA YARDIMCILARI.

INSERT ve PLACE gibi KARAKTER DİZİSİ İŞLEME komutları, dize işleme konusunda size tam kontrol sağlar.

Ekran biçimlendirmeyi kolaylaştırmak için CENTRE ve PRINT AT gibi TEXT komutları.

Klavyeden yazılanlar üzerinde tam kontrol sahibi olmanızı sağlamak için FETCH ve INKEY gibi GELİŞTİRİLMİŞ GİRİŞ komutları.

Tam sayı bölme işlemini daha basit hale getirmek için MOD ve DIV gibi EKSTRA ARİTMETİK OPERATÖRLER.

Sayısal dönüştürme komutları, ikili veya onaltılık sayıları ondalık eşdeğerlerine dönüştürmenizi sağlar.

Daha okunaklı kod yazmanızı sağlayacak PROC ve IF..THEN..ELSE gibi YAPILANDIRILMIŞ PROGRAMLAMA komutları.

SCRSV ve COPY gibi EKRAN MANİPÜLASYONU yardımcıları, ekran verilerini depolamanıza/yüklemenize ve/veya yüksek/düşük çözünürlüklü bir ekranın çıktısını almanıza olanak tanır.

Ekranda şekiller çizmenizi sağlayan CIRCLE ve PAINT gibi GRAFİK ÇİZİM komutları.

YARATIK ve KULLANICI TANIMLI GRAFİK komutları, örneğin TASARIM, MOB AYARLA, ALGILA ve KONTROL, kendi 'hareketli nesne bloklarınızı' oluşturmanıza ve canlandırmanıza veya kendi grafik karakterlerinizi tasarlamanıza olanak tanır.

Ses efektleri oluşturmanıza, müzik bestelemenize ve çalmanıza olanak sağlayan WAVE ve ENVELOPE gibi MUSIC komutları.

Dosya işlemeyi basitleştirmek için DIR gibi DİSK İŞLETİM komutları.

SIMON BASIC kartuşunun sağladığı komut yelpazesi, COMMODORE 64'ünden en iyi şekilde yararlanmak isteyen herkes için onu vazgeçilmez bir araç haline getiriyor.

Bu kılavuz, COMMODORE 64'te BASIC programlamayı öğretmek için tasarlanmamıştır. BASIC programlama hakkında hiçbir bilginiz yoksa, lütfen aşağıdakilerden birine bakın:

COMMODORE 64 Kullanıcı Kılavuzu (bilgisayarınızla birlikte verilir)

Andrew Colin'in BASIC Bölüm 1 ve 2'ye Giriş kitabı.

1.2 SIMON BASIC KILAVUZU

Bu kılavuz aşağıda belirtildiği gibi on üç bölüme ayrılmıştır:

BİRİNCİ BÖLÜM-SIMON BASIC'E GİRİŞ

Bu bölüm, SIMON BASIC'i genel hatlarıyla açıklamaktadır. Ayrıca, kartuşun nasıl doldurulacağını ve bir SIMON BASIC komutunun nasıl girileceğini de açıklamaktadır. Bu kılavuzda her komutu açıklamak için kullanılan kurallar da yer almaktadır. SIMON BASIC'in standart COMMODORE 64 BASIC ile uyumluluğu da ele alınmaktadır. SIMON BASIC programlarının nasıl saklanacağı, yükleneceği ve çalıştırılacağına dair talimatlar da verilmektedir.

İKİNCİ BÖLÜM-PROGRAMLAMA YARDIMCILARI

Bu bölümde, daha hızlı ve daha verimli BASIC programlamayı kolaylaştırmak için AUTO ve TRACE gibi komutlar yer almaktadır. Ayrıca, COMMODORE 64'ün fonksiyon tuşlarının programlanmasını sağlayan KEY komutu da bu bölümde yer almaktadır.

ÜÇÜNCÜ BÖLÜM-GİRİŞ DOĞRULAMA VE METİN İŞLEME

Bu bölüm, karakter dizisi işlemeyi iyileştirmek için INSERT ve PLACE gibi komutları içerir. Ayrıca, kullanıcı girdisi üzerinde gelişmiş kontrol sağlayan FETCH ve INKEY komutları da yer almaktadır. Ayrıca, CENTRE ve PRINT AT gibi ekran metni biçimlendirme komutları da açıklanmaktadır.

DÖRDÜNCÜ BÖLÜM-SAYISAL OLMAYAN YARDIMCILAR

Burada üç ek aritmetik operatör olan MOD, DIV ve FRAC açıklanmaktadır. İlk iki komut tamsayı bölme işlemiyle ilgilenirken, üçüncü komut bir sayının kesirli kısmının çıkarılmasını sağlar. Bu bölüm ayrıca, ikili veya onaltılık sayıları ondalık biçime dönüştürmek için kullanılan % ve $ komutlarının ve ek bir Boole işlemi gerçekleştiren EXOR komutunun açıklamasını da içermektedir.

BEŞİNCİ BÖLÜM-DİSKET KOMUTLARI

Burada iki komut ele alınmaktadır: DISK ve DIR. DISK, biçimlendirme ve dosya silme gibi çeşitli disk çalıştırma komutlarının tek bir komutla yapılmasını sağlar; yani görev tamamlandığında disk kanalı otomatik olarak kapatılır. DIR ise bir disket dizininin tamamının veya seçili bir bölümünün ekranda görüntülenmesini sağlar.

ALTINCI BÖLÜM-SIMON BASIC İLE GRAFİKLER

Bu bölümde, SIMON BASIC grafik çizim komutlarının geniş yelpazesi açıklanmaktadır. Bu komutlar, ekranda şekiller çizmenize ve bunları COMMODORE 64 tarafından sağlanan on altı renkten herhangi biriyle boyamanıza olanak tanır.

YEDİNCİ BÖLÜM-EKRAN İŞLEMLERİ

Bu bölüm, ekranın herhangi bir alanının herhangi bir yönde nasıl kaydırılacağını açıklamaktadır. Ayrıca, ekranın bir alanını başka bir konuma taşıma, ekran karakterlerinin rengini değiştirme ve ekran verilerini depolama ve geri çağırma komutları da yer almaktadır. Yüksek/düşük çözünürlüklü ekranların yazdırılmasını sağlayan komutlar da açıklanmaktadır.

SEKİZİNCİ BÖLÜM-SPRITE VE KULLANICI TANIMLI GRAFİKLER

Sekizinci Bölüm, COMMODORE 64 Sprite grafiklerinin tasarımı ve animasyonuyla ilgili SIMON BASIC komutlarını açıklamaktadır. Ayrıca, kendi grafik karakterlerinizi oluşturmanıza olanak tanıyan talimatlar da yer almaktadır.

DOKUZUNCU BÖLÜM-YAPILANDIRILMIŞ PROGRAMLAMA

Burada SIMON BASIC'in çeşitli yapılandırılmış programlama komutları açıklanmaktadır.

ONUNCU BÖLÜM-HATA TUTMA

Onuncu Bölüm, programlarınızın çökmesini önlemek için belirli BASIC program hatalarının yakalanmasını sağlayan komutlar içerir.

ON BİRİNCİ BÖLÜM-SIMON BASIC İLE MÜZİK YAPMAK

Burada, COMMODORE 64'te müzik çalmanızı sağlayan SIMON BASIC komutları açıklanmaktadır.

ON İKİNCİ BÖLÜM-FONKSİYONLARI OKUYUN

Bu bölüm, oyun kolu gibi bir oyun cihazıyla kontrolü bir programa entegre etmenizi sağlayan PENX ve POT gibi fonksiyonları açıklamaktadır.

ON ÜÇÜNCÜ BÖLÜM-SIMON BASIC PROGRAMLARINA ÖRNEKLER

On Üçüncü Bölüm, kartuşla neler başarılabileceğinin gösterilmesi amacıyla SIMON BASIC kullanılarak yazılmış programların listelerini içermektedir.

EK A-HATA MESAJLARI

SIMON BASIC komutlarını kullanırken karşılaşabileceğiniz hata mesajlarının listesi ve olası nedenleri bu Ek'te verilmiştir.

SÖZLÜK

Bu kılavuzda kullanılan terimlerin listesi ve tanımları bu bölümde verilmiştir.

1.3 SIMON BASIC'E BAŞLAMAK

SIMON BASIC kartuşu her zaman COMMODORE 64'e güç KAPALI durumdayken takılmalı veya çıkarılmalıdır. Kartuş, etiketi üstte olacak şekilde bilgisayarın arkasındaki kartuş yuvasına takılır. (COMMODORE 64 Kullanım Kılavuzuna bakın.)

SIMON BASIC'i kullanmaya başlamak için, kartuşu yerindeyken bilgisayarı açmanız yeterlidir. Ardından aşağıdaki mesaj görüntülenir:

```
*** EXPANDED CBM V2 BASIC ***

30719 BYTES FREE

READY.
```

Tüm SIMON BASIC komutları artık COMMODORE 64 işletim sisteminize dahil edilmiştir ve diğer BASIC komutları gibi istediğiniz zaman kullanılabilir. SIMON BASIC'in COMMODORE 64 belleğinin yaklaşık 8K'sını kullandığını unutmayın.

1.4 SIMON BASIC KOMUTLARI

Aşağıda SIMON BASIC kartuşu tarafından COMMODORE 64 işletim sisteminize eklenen komutların listesi yer almaktadır:

Programlara giriş, hata ayıklama, listeleme ve güvenlik komutları:

KEY, DISPLAY, AUTO, RENUMBER, PAUSE, MERGE, PAGE, OPTION, DELAY, FIND, TRACE, RETRACE, DUMP, COLD, OLD, RESET, CGOTO, DISAPA, SECURE.

Metin düzenleme, ekran biçimlendirme ve giriş doğrulama komutları:

INSERT, INST, PLACE, DUP, USE, CENTRE, AT, LIN, FETCH, INKEY, ON KEY, DISABLE, RESUME.

Tam sayı bölme, sayısal dönüştürme ve ek bir Boole işlemi için komutlar.

MOD, DIV, FRAC, %, $, EXOR.

Disket işleme komutları:

DISK, DIR.

Grafik çizimi için komutlar:

COLOUR, HIRES, MULTI, NRM, HICOL, LOW COL, PLOT, LINE, REC, CIRCLE, ARC, ANGL, BLOCK, PAINT, NRM, DRAW, ROT, CHAR, TEXT, TEST, CSET.

Ekran verilerini depolama, yazdırma ve düzenleme komutları:

EFT/RIGHT/UP/DOWN scrolling, BCKGNDS, FLASH, OFF, BFLASH, FCHR, FCOL, FILL, MOVE, INV, SCRSV, SCRLD, COPY, HRDCPY.

Yaratıkları oluşturma/canlandırma ve kendi karakterlerinizi yaratma komutları:

DESIGN, @, CMOB, MOB SET, MMOB, RLOCMOB, DETECT, CHECK, MOB OFF, MEM.

Yapılandırılmış programlama komutları:

IF, THEN, ELSE, REPEAT, UNTIL, LOOP, EXIT IF, END LOOP, PROC, CALL, EXEC, END PROC, RCOMP, LOCAL, GLOBAL, NO ERROR, ON ERROR, OUT.

Müzik sentezi için komutlar:

WAVE, ENVELOPE, MUSIC, VOL, PLAY, SOUND.

Oyun cihazlarını programlarınızla birlikte kullanmak için işlevler:

PENX, PENY, POT, JOY.

1.5 KOMUT GİRİŞİ

Tüm SIMON BASIC komutları, standart Commodore BASIC komutlarıyla aynı şekilde girilir. Çoğu SIMON BASIC komutu doğrudan modda veya bir programın parçası olarak kullanılabilir. Bu kuralın istisnaları, kılavuzun her bölümünün girişinde belirtilmiştir.

1.6 KURALLAR

Bu kılavuzdaki her SIMON BASIC komutunun biçimi aşağıdaki gösterim yöntemi kullanılarak sunulmuştur:

1. Parantez içindekiler ve büyük harflerle yazılanlar tam olarak gösterildiği gibi yazılmalıdır.

2. Küçük harflerle basılan öğeler, kullanıcı tarafından sağlanan veya değişken bir girişi, örneğin koordinatları veya çizim rengini gösterir.

3. Tırnak işareti ve virgül gibi diğer simgeler gösterildiği gibi tam olarak yazılmalıdır.

4. RETURN tuşuna basıldığında **<RETURN>** gösterilir.

5. Alfabetik ve sayısal karakterler dışındaki tuşlar, listede <> işaretiyle çevrelenmiş tuş yüzeyindeki adlarıyla gösterilir, örneğin <CLR/HOME>. Bunlar ekranda ters karakterler olarak görünür. İki tuş çevrelenmişse, örneğin <CTRL RVS ON>, ikinci tuşa basmadan önce ilk tuşu basılı tutmanız gerekir.

6. FIND komutu hariç (bkz. Bölüm 2.10), tüm SIMON BASIC anahtar sözcükleri komutun ilk parametresinden bir boşlukla ayrılmalıdır.

İKİNCİ BÖLÜM
PROGRAMLAMA YARDIMCILARI

2.1 GİRİŞ

SIMON BASIC, BASIC programlarınıza girerken, hata ayıklarken ve listelerken, SIMON BASIC komutlarını içerip içermediğine bakılmaksızın kullanışlı olan çeşitli komutlar sağlar.

KEY komutu, COMMODORE 64'ün fonksiyon tuşlarının programlanmasını sağlar. DISPLAY, bu tuşlara atanmış değerleri listeler. AUTO ve RENUMBER komutları, otomatik program satırı numaralandırması oluşturur. MERGE, depolanan bir BASIC programını, COMMODORE 64'ün belleğinde bulunan programla birleştirir.

PAGE komutu, ekranda programları listelerken kaç ekran satırı kullanmak istediğinizi belirtmenizi sağlar. OPTION komutu, program listesindeki tüm SIMON BASIC komutlarını vurgular. DELAY komutu, ekrandaki program listelerinin kaydırma hızını kontrol etmenizi sağlar.

TRACE ve RETRACE komutları, çalıştırıldıkları sırada program satırlarının numaralarını görüntüler. DUMP komutu, dizi dışı tüm değişkenlerin değerlerini listeler. FIND, belirli bir karakter dizisinin tüm örneklerini bulur.

PAUSE komutu, programınızda bir zaman gecikmesi ayarlamak için kullanılır. CGOTO, hesaplanmış bir satır numarasına geçer. RESET, COMMODORE 64'e tanımlanmış bir program satırından veri okumasını söyler. SECURE ve DISAPA komutları, yetkisiz kişilerin kodunuzu incelemesini önlemek için belirtilen program satırlarını 'boş' bırakır. COLD, COMMODORE 64'ü SIMON BASIC başlangıç ekranına döndürür. OLD komutu, YENİ olarak ayarlanmış bir programı kurtarmanıza olanak tanır.

Bu bölümdeki tüm komutların doğrudan modda veya bir programın parçası olarak kullanılabileceğini unutmayın.

2.2 FONKSİYON TUŞLARINA KOMUT ATAMA

2.2.1 KEY

FORMAT: KEY numarası,"kod"

AMAÇ: Bir fonksiyon tuşuna komut atamak.

KEY, COMMODORE 64 fonksiyon tuşlarına kendi komutlarınızı atamanızı ve ardından dilerseniz bu komutları değiştirmenizi sağlar. Komut formatındaki sayı, kullanmak istediğiniz 1 ile 16 arasında fonksiyon tuşunu gösterir. İkinci parametre, bu tuşa atamak istediğiniz koddur. Her tuşa en fazla on beş karakter atanabilir. Tuşlara normal şekilde basıldığında F1, F3, F5 ve F7 fonksiyonları elde edilir. SHIFT tuşunu basılı tutup aynı tuşlara basıldığında F2, F4, F6 ve F8 fonksiyonları elde edilir. Commodore logosu tuşunu basılı tutup tuşlara basıldığında F9, F10, F11 ve F12 fonksiyonları elde edilir. SHIFT tuşunu ve Commodore logosu tuşunu basılı tutarsanız F13, F14, F15 ve F16 fonksiyonları elde edilir. Her tuşa atadığınız kodun tırnak işareti içine alınması gerektiğini unutmayın.

ÖRNEK: MOB SET komutunu F8 fonksiyon tuşuna atamak için:

KOMUT: `KEY 8,"MOB SET"` **<RETURN>**

SONUÇ: SIMON BASIC kodu MOB SET artık F8 fonksiyon tuşuna atanmıştır ve bu tuşa her basıldığında görüntülenecektir.

2.2.2 SATIR BAŞI EKLEME

Bir fonksiyon tuşu komutunun ardından RETURN tuşuna basma gereksinimini ortadan kaldırmak için, tuş atamasına aşağıdaki gibi bir satır sonu ekleyebilirsiniz

a) Komutunuzu tuşa atayın (Bölüm 2.2.1'e bakın). Tırnak işaretlerini yazın ancak RETURN tuşuna basmayın.

b) Tür + CHR$(13) yazın ve RETURN tuşuna basın.

Artık fonksiyon tuşuna bastığınızda, atanan komutu takiben otomatik olarak RETURN üreteceksiniz.

ÖRNEK: F7 fonksiyon tuşuna BASIC komutu LIST ve otomatik satır başı atamak için:

KOMUT: `KEY 7,"LIST" + CHR$(13)` **<RETURN>**

SONUÇ: Artık F7 tuşuna basarak bir programı listeleyebilirsiniz.

2.2.3 DISPLAY

FORMAT: DISPLAY

AMAÇ: Fonksiyon tuşlarına atanan komutları listelemek.

DISPLAY, mevcut fonksiyon tuşu atamalarını incelemenizi sağlar.

ÖRNEK: Önceki iki bölümdeki atama örneklerinden sonra fonksiyon tuşu atamalarını listelemek için:

KOMUT: `DISPLAY <RETURN>`

EKRAN:

```
KEY 1 ""
KEY 2 ""
KEY 3 ""
KEY 4 ""
KEY 5 ""
KEY 6 ""
KEY 7 "LIST" + CHR$(13)
KEY 8 "MOB SET"
KEY 9 ""
KEY 10 ""
KEY 11 ""
KEY 12 ""
KEY 13 ""
KEY 14 ""
KEY 15 ""
KEY 16 ""
```

2.3 AUTO

FORMAT: AUTO başlangıç satır numarası, artış

AMAÇ: Program satır numaralarını belirtilen bir artışla otomatik olarak oluşturmak.

AUTO komutu girildiğinde, tanımladığınız başlangıç programı satır numarası görüntülenir ve ardından gelen imleç, bir satır kod girişi bekler. Daha sonra, her satır kod yazıp RETURN tuşuna bastığınızda, belirttiğiniz artış bir önceki satırın numarasına eklenir. Elde edilen değer, bir sonraki program satır numarası olarak görüntülenir. Bu işlevi sonlandırmak için, satır numarası görüntülendiğinde RETURN tuşuna basmanız yeterlidir.

ÖRNEK: Program satır numaralarını 10. satırdan başlayarak 5'erli aralıklarla otomatik olarak oluşturmak için:

KOMUT:
```
AUTO 10,5 <RETURN>
```

EKRAN:
```
10
```

YAZ:
```
GET A$ <RETURN>
```

EKRAN:
```
10 GET A$
15
```

YAZ:
```
IF A$ = "" THEN 10 <RETURN>
```

EKRAN:
```
10 GET A$
15 IF A$ = "" THEN 10
20
```

SONUÇ: Her seferinde bir satır kod girip RETURN tuşuna bastığınızda, bir öncekinden 5 büyük satır numarası görüntülenir.

ÖRNEK: Yukarıda listelenen programda otomatik program satır numaralandırma sonlandırmak için:

YAZ:
```
<RETURN>
```

KOMUT:
```
LIST <RETURN>
```

EKRAN:
```
10 GET A$
15 IF A$ = "" THEN 10
READY.
```

SONUÇ: Otomatik numaralandırma sonlandırıldı.

2.4 RENUMBER

FORMAT: RENUMBER başlangıç satır numarası, artış

RENUMBER, tüm program satırlarının numaralarını otomatik olarak değiştirir. Program artık belirttiğiniz başlangıç satır numarasından başlar ve sonraki tüm satır numaraları seçilen artışla görüntülenir. Bu komut, özellikle programda daha fazla kod eklemek için alana ihtiyacınız varsa kullanışlıdır.

NOT

RENUMBER komutu GOTO'ları veya GOSUB'ları yeniden numaralandırmaz. Ancak SIMON BASIC, bu talimatları yapılandırılmış programlama komutlarıyla değiştirerek bu talimatlara olan ihtiyacı ortadan kaldırır. Bkz. Bölüm 9.

ÖRNEK: Aşağıdaki programın tüm program satırlarını yeniden numaralandırmak için:

GİR:
```
1 PRINT"␣"
2 FOR X = 1 TO 20
3 Z = RND(1) * 255
4 POKE 53280,Z
5 FOR Y = 1 TO 250: NEXT Y,X
```

KOMUT: `RENUMBER 100,10` **<RETURN>**

YAZ: `LIST` **<RETURN>**

EKRAN:
```
100 PRINT"␣"
110 FOR X = 1 TO 20
120 Z = RND(1) * 255
130 POKE 53280,Z
140 FOR Y = 1 TO 250: NEXT Y,X
```

2.5 PAUSE

FORMAT: PAUSE "mesaj", saniye sayısı

Veya: PAUSE saniye sayısı

AMAÇ: Programın belirli bir süre boyunca yürütülmesini durdurmak için.

PAUSE, bir programın çalışmaya devam etmeden önce beklemesine neden olur. Aralık, saniyelerle ölçülen önceden belirlenmiş bir süredir. Saniyenin kesirlerinin KULLANILAMAYACAĞINI unutmayın. PAUSE komutu, mesajlı veya mesajsız olmak üzere iki şekilde kullanılabilir.

PAUSE komutuna tırnak işaretleri arasında bir mesaj eklenirse, mesaj belirtilen süre boyunca görüntülenir. RETURN tuşuna basıldığında duraklama kesilir ve program yürütme işlemi devam eder.

PAUSE'dan sonra herhangi bir mesaj eklenmezse, program belirtilen gecikme süresi geçene kadar bekler.

ÖRNEK: 10 saniyelik bir program gecikmesine neden olmak için:

GİR:
```
100 PAUSE 10
```

SONUÇ: Programın 100. satırına gelindiğinde 10 saniyelik bir gecikme meydana gelir.

ÖRNEK: Bir mesajı görüntülemek ve 1 dakika beklemek için:

GİR:
```
100 PAUSE "DEVAM ETMEK ICIN RETURN'E BAS
IN",60
```

SONUÇ: Bu satır çalıştırıldığında, DEVAM ETMEK İÇİN RETURN'E BASIN görüntülenir ve program bir dakika boyunca veya RETURN tuşuna basılana kadar devam etmez.

2.6 CGOTO

FORMAT: CGOTO ifade

Veya: CGOTO işlenen operatör değişkeni

AMAÇ: Programın dallanacağı satır numarasını hesaplamak.

CGOTO komutu, bir hesaplamanın sonucuna göre belirlenen değişken satır numarasına dallanmanızı sağlar.

ÖRNEK: Bir döngü değişkeni tarafından belirtilen beş farklı satır numarasına dallanmak için:

GİR:
```
10 REM "*** CGOTO'YA ORNEK ***"
20 FOR I = 1 TO 5
30 CGOTO I * 10 + 40
40 END
50 PRINT"I = 1":NEXT
60 PRINT"I = 2":NEXT
70 PRINT"I = 3":NEXT
80 PRINT"I = 4":NEXT
90 PRINT"I = 5":NEXT
```

SONUÇ: Her I değeri için satır numarası hesaplanır ve o satır çalıştırılır.

2.7 RESET

FORMAT: RESET satır numarası

AMAÇ: Veri işaretçilerini belirli bir veri satırına taşımak.

Standart BASIC'te veriler her zaman sıralı olarak okunur, yani ilk veri öğesi ilk READ ifadesi tarafından, ikinci öğe bir sonraki ifade tarafından kullanılır, vb. RESET, veri bloğu içindeki okumanın başlayacağı program satırını belirtmenizi sağlar, yani programdaki ilk veri öğesinden başlamanıza gerek yoktur veya belirli bir noktaya kadar bazı öğeleri atlayabilirsiniz.

ÖRNEK: Kullanıcı girdisine bağlı olarak belirli verileri seçmek için:

GİR:

```
10 REM"*** RESET'E ORNEK ***"
20 PRINT"□"
30 PRINT"HANGI KATEGORI?":PRINT:PRINT
40 PRINT"1) KOPEKLER","2) KEDILER","3) K
USLAR","4) BALIKLAR"
50 INPUT A: IF A < 1 OR A > 4 THEN PRINT
"□□": GOTO 50
60 IF A = 1 THEN RESET 100
70 IF A = 2 THEN RESET 110
80 IF A = 3 THEN RESET 120
90 IF A = 4 THEN RESET 130
95 FOR I = 1 TO 5
97 READ A$:PRINT A$:NEXT I
99 PAUSE 10:GOTO 20
100 DATA "KURT","BULDOG","PITBUL","LABRA
DOR","KANGAL"
110 DATA "VAN KEDISI","ANKARA KEDISI","T
EKIR","KARAYAK","KORAT"
120 DATA "KARTAL","SAHIN","DOGAN","SERCE
","BULBUL"
130 DATA "HAMSI","PALAMUT","ISTAVRIT","L
UFER","MEZGIT"
```

YAZ: `RUN` **<RETURN>**

GİR: `3` **<RETURN>**

EKRAN:

```
KARTAL
SAHIN
DOGAN
SERCE
BULBUL
```

SONUÇ: Program, kullanıcı girdi değerine ilişkin satır numarasından başlayarak beş adet veriyi okur.

EYLEM: **<RUN/STOP>** tuşunu basılı tutun ve **<RESTORE>** tuşuna basın.

SONUÇ: Program durur.

2.8 MERGE

FORMAT: MERGE "program ismi",cihaz numarası

AMAÇ: Daha önce kaydedilmiş bir programı yüklemek ve onu COMMODORE 64'ün belleğinde bulunan programa dahil etmek.

Aygıt numarası, MERGE edilecek programın depolandığı çevre biriminin numarasını belirtir. Bu sayı, kaset birimi için 1, disk birimi için 8'dir. Aygıt numarası belirtilmezse, 1, yani kaset olduğu varsayılır. Program adı, BASIC LOAD komutunda olduğu gibi belirtilir.

DİKKAT

BİRLEŞTİRİLEN PROGRAM, HAFIZADAKİ PROGRAMI TAKİP EDECEKTİR, yani BİRLEŞTİRİLEN PROGRAM SATIRLARI ARALARINA EKLENMEK YERİNE EKLENECEKTİR. BİRLEŞTİRİLEN PROGRAMI ÇALIŞTIRMADAN ÖNCE YENİDEN NUMARALANDIRMAK İÇİN YENİDEN NUMARALANDIRMA KOMUTUNU (bkz. Bölüm 2.4) KULLANIN.

ÖRNEK: "SIMON BASIC1" adlı kaset programını şu anda bellekte bulunan programla BİRLEŞTİRMEK için:

EYLEM: Küçük bir program yazın ve bunu "SIMON BASIC1" adı altında kasete kaydedin.

KOMUT: `NEW` **<RETURN>**

EYLEM: Başka bir küçük program yazın.

KOMUT: `MERGE "SIMON BASIC1",1` **<RETURN>**

EKRAN:
```
PRESS PLAY ON TAPE
```

EYLEM: Kaset ünitesindeki PLAY düğmesine basın.

EKRAN:
```
LOADING SIMON BASIC1
READY.
```

SONUÇ: İki program artık birleştirildi.

2.9 PROGRAM LİSTELEME YARDIMCILARI

2.9.1 PAGE

FORMAT: PAGE n

AMAÇ: Bir program listesini n satırlık 'sayfalara' bölmek.

PAGE, bir programı listelerken kullanmak istediğiniz ekran satırı sayısını belirtmenize olanak tanır. Komut yürütüldüğünde, bir LIST, programın ilk satır numarasını görüntüler. Listenin her bölümü daha sonra RETURN tuşuna basılarak görüntülenebilir. Sıfır parametresi, programın normal şekilde listelenmesini sağlayarak sayfalamayı sonlandırır. Bu komuttaki parametrenin, birden fazla ekran satırını kaplayabilen program satırlarını değil, ekran satırı sayısını ifade ettiğini unutmayın. Bir program satırı tanımladığınız ekran sınırlarını aşarsa, o satırın tamamı bir sonraki ekranda görünür.

ÖRNEK: Sadece 5 ekran satırı kullanarak bir programı listelemek için:

EYLEM: Ondan fazla satır kod içeren bir program yükleyin veya oluşturun.

KOMUT: `PAGE 5` **<RETURN>**

YAZ: `LIST` **<RETURN>**

SONUÇ: İlk program satır numarası görüntülenir.

KOMUT: **<RETURN>** tuşuna basın.

SONUÇ: Programınızın ilk 5 satırı görüntülenir.

KOMUT: **<RETURN>** tuşuna basın.

SONUÇ: Programınızın ikinci 5 satırı görüntülenir.

KOMUT: `PAGE 0` **<RETURN>**

YAZ: `LIST` **<RETURN>**

SONUÇ: Programınız normal şekilde listeleniyor.

2.9.2 OPTION

FORMAT: OPTION n

AMAÇ: Bir program listelendiğinde tüm SIMON BASIC komutlarını vurgulamak.

10 parametreli OPTION komutu, program ekranda veya yazıcıda listelendiğinde tüm SIMON BASIC komutlarının ters alanda vurgulanmasına neden olur. 10'dan farklı bir parametre (0 ile 255 arasında) vurgulamayı kapatır.

DİKKAT

OPTION KOMUTU KULLANILDIKTAN SONRA YAZDIRILAN LİSTELER, YAZICI ŞERİDİNİZİN ÇOK HIZLI BİR ŞEKİLDE ESKİMESİNE NEDEN OLABİLİR. BU NEDENLE BU TÜR LİSTELERİN SIK SIK YAZDIRILMAMASI TAVSİYE EDİLİR.

Aşağıdaki örnek programda kullanılan bazı komutların henüz ele alınmadığını unutmayın. Bunlar yalnızca OPTION komutunun kullanımını göstermek amacıyla eklenmiştir.

ÖRNEK: Aşağıdaki programdaki tüm SIMON BASIC komutlarını vurgulamak için:

```
10 HIRES 0,1
20 CIRCLE 160,100,80,60,1
30 REC 80,40,160,120,1
40 PAUSE 10
50 CSET 0 : END
```

KOMUT: `OPTION 10` **<RETURN>**

YAZ: `LIST` **<RETURN>**

SONUÇ: SIMON BASIC komutları ekran listesinde ters alanda vurgulanmıştır.

KOMUT: `OPTION 0` **<RETURN>**

YAZ: `LIST` **<RETURN>**

SONUÇ: Program normal şekilde listeleniyor.

NOT

Yazıcıya listeleme yaparken tüm komutları tek bir satıra girin, örneğin:

```
OPEN 4,4:CMD4:LIST:PRINT#4:CLOSE4
```
<RETURN>

Sonraki herhangi bir programı listelemeden önce yazıcıyı kapatıp tekrar açın.

2.9.3 DELAY

FORMAT: DELAY n

AMAÇ: Bir program listesinin kaydırma hızını değiştirmek.

Bir program listeleme sırasında SHIFT tuşu basılı tutulduğunda, ekran kaydırma hızı yavaşlar. DELAY komutu, bu yavaşlatılmış listelemenin hızını değiştirir. Komutu takip eden parametre, gecikmenin süresini belirler. Bu sayı 1 ile 255 arasında olmalıdır. Komuttaki daha büyük bir değer, program listeleme kaydırma hızının orantılı olarak daha yavaş olmasına neden olur.

ÖRNEK: Mevcut en yavaş hızdaki bir programı listelemek için:

KOMUT: `DELAY 255` **<RETURN>**

YAZ: `LIST` **<RETURN>**

EYLEM: **<SHIFT>** tuşunu basılı tutun.

SONUÇ: Program listesi karakter karakter görüntülenir.

KOMUT: **<SHIFT>** tuşunu bırakın.

SONUÇ: Program normal şekilde listeleniyor.

NOT

COMMODORE 64'te herhangi bir BASIC programını listelerken, CTRL tuşu bırakılana kadar ekran kaydırma hızını yavaşlatır.

2.10 FIND

FORMAT: FINDkod

Veya: FINDkarakter dizesi

AMAÇ: Bir BASIC programında belirli bir kod veya karakter dizisini aramak ve bu kodun göründüğü program satırlarının numaralarını görüntülemek.

FIND, bir BASIC programında belirli bir kod veya karakter dizisi örneğini bulmak için kullanılır. Komut, dizeyi veya kodu içeren tüm satır numaralarını görüntüler. FIND ile belirtilen karakterler veya son karakter ile RETURN arasındaki boşlukların, arama yapılan karakter dizisinin bir parçası olarak kabul edildiğini unutmayın. Bu nedenle, program anahtar sözcükleri öncesinde boşluk OLMADAN girilmelidir.

ÖRNEK: Aşağıdaki programda ABCD karakter dizisini bulmak için:

```
10 REM FIND ABCD
20 REM PRINT "ABCD" DIKEY OLARAK
30 PRINT "ABCD DIKEY OLARAK"
40 A$ = "ABCD"
50 FOR C = 1 TO LEN (A$)
60 PRINT MID$(A$,C,1):NEXT
70 REM ABCD TAMAM
```

KOMUT: `FIND"ABCD"` **<RETURN>**

EKRAN:
```
20  40
```

SONUÇ: Tırnak işaretleri arasında ABCD karakter dizisini içeren her program satır numarası görüntülenir. 10 ve 70 numaralı satırların görüntülenmediğini unutmayın çünkü ABCD bu satırlarda tırnak işaretleri içinde değildir.

2.11 PROGRAM HATA AYIKLAMA YARDIMCILARI

2.11.1 TRACE

FORMAT: TRACE n

AMAÇ: Yürütülen program satırının numarasını görüntülemek için.

TRACE komutu, bir program çalıştırılmadan önce girilir. Komut parametresi olarak 10 değeri kullanılırsa, programı çalıştırdığınızda ekranın sağ üst köşesinde bir "pencere" görüntülenir. Program satırları çalıştırıldıkça, sayılar pencerede görüntülenir. Herhangi bir anda en fazla altı sayı gösterilir. Biçimi şu şekildedir: # (satır numarası). Penceredeki satırlar otomatik olarak kaydırılır ve çalıştırılan sondan bir önceki program satır numarası pencerenin alt kısmında görünür. Commodore logosu tuşu basılı tutulduğunda, programda satır satır gezinmenizi sağlar. 0 parametresi TRACE'i kapatır. TRACE komutunun yüksek çözünürlüklü bir ekranda veya MEM komutu (bkz. Bölüm 8.3.2) kullanılmışsa KULLANILAMAYACAĞINI unutmayın.

DİKKAT

İZLEME PENCERESİ, EKRANDA GÖRÜNTÜLENEN HER ŞEYİN ÜZERİNE YAZAR. BU NEDENLE, GÖZLEMLEMEK İSTEDİĞİNİZ HERHANGİ BİR METNİN BU ALANIN DIŞINA YAZDIRILMASINA DİKKAT EDİN.

ÖRNEK: Aşağıdaki program ÇALIŞTIRILDIĞINDA program satır numaralarını tek tek görüntülemek için:

```
10 PRINT "<SHIFT CLR/HOME>"
20 FOR X = 65 TO 96
30 PRINT "<CLR/HOME>";CHR$(X)
40 FOR Z= 1 TO 250: NEXT Z
50 NEXT X
60 GOTO 20
```

KOMUT: `TRACE 10` **<RETURN>**

YAZ: `RUN` **<RETURN>**

SONUÇ: Programın her satırı çalıştırıldığında pencerede görünür.

YAZ: `TRACE 0 RUN` **<RETURN>**

SONUÇ: Pencere kaybolur ve program normal şekilde çalışır.

2.11.2 RETRACE

FORMAT: RETRACE

AMAÇ: Bir programı düzenledikten sonra İZLEME'yi sürdürmek.

TRACE komutunu kullanırken, program yürütmeyi durdurup ekranı temizlerseniz, TRACE penceresi kaybolur. RETRACE komutu, TRACE'i tekrar açar ve program durdurulmadan önce yürütülen son satır numaralarını görüntüler. Program yeniden çalıştırıldığında, normal TRACE ekranı görüntülenir. Yürütme, programın durdurulduğu yerden değil, başlangıcından devam eder. TRACE komutu kapatılmışsa RETRACE'in kullanılamayacağını unutmayın.

ÖRNEK: Önceki bölümdeki programı kullanarak, programın yürütülmesini durdurmak için ekranı temizleyin ve TRACE ile yeniden çalıştırın:

KOMUT: `TRACE 10` **<RETURN>**

YAZ: `RUN` **<RETURN>**

SONUÇ: Programın her satırı çalıştırıldığında TRACE penceresinde görünür.

EYLEM: **<RUN/STOP>** tuşuna basın.

YAZ: `LIST` **<RETURN>**

GİR: `30 PRINT "[SHIFT+CLR/HOME]";CHR$(X),X` **<RETURN>**

EYLEM: **<SHIFT>** tuşunu basılı tutun ve **<CLR/HOME>** tuşuna basın.

SONUÇ: Ekran temizlenir.

KOMUT: `RETRACE` **<RETURN>**

SONUÇ: Ekranın sağ üst köşesindeki pencere tekrar belirir ve program durdurulduğunda gösterilen satır numaralarını görüntüler.

EYLEM: İmleci pencerenin altına taşıyın.

YAZ: `RUN` **<RETURN>**

SONUÇ: Program çalışırken satır numaraları tekrar TRACE penceresinde görüntülenir.

2.12 DUMP

FORMAT: DUMP

AMAÇ: Dizi dışı tüm değişkenlerin değerlerini görüntülemek.

DUMP komutu, dizilerde bulunanlar hariç tüm değişkenlerin değerlerini görüntüler. Gösterilen değerler, program **<RUN/STOP>** tuşuna basılarak veya bir program sonlandırıcıya ulaşılarak durdurulduğunda değişkenlerde bulunan değerlerdir. Değişkenler, programda tanımlandıkları sırayla listelenir ve şu biçimde görüntülenir:

değişken adı = değer

NOT

Programınız 25'ten fazla değişken içeriyorsa, listenin ekrandan kaymasını önlemek için **<CTRL>** tuşunu basılı tutun. Listenin geri kalanını görüntülemek için tuşu bırakın.

ÖRNEK: Aşağıdaki programdaki değişkenleri görüntülemek için:

```
10 A$ = "RASTGELE RENKLER"
20 PRINT "■",A$
30 X = INT(RND(8) * 15)
60 POKE 53281,X
70 FOR C = 1 TO 100:NEXT C
80 GOTO 30
```

YAZ: **RUN <RETURN>**

EYLEM: Ekran birkaç kez renk değiştirdikten sonra **<RUN/STOP>** tuşunu basılı tutun ve **<RESTORE>** tuşuna basın.

KOMUT: **DUMP <RETURN>**

EKRAN:

```
A$ = "RASTGELE RENKLER"
X = 9
C = 80
```

(X ve C değerlerinin rastgele üretildiğini ve bu iki değişken için gösterilen sayıların programın ne zaman durdurulduğuna bağlı olacağını unutmayın.)

2.13 COLD

FORMAT: COLD

AMAÇ: COMMODORE 64'ü SIMON BASIC'in başına sıfırlamak için:

COLD, COMMODORE 64'ün belleğinde tutulan tüm programları temizler ve SIMON BASIC kartuşu takılıyken bilgisayarı açtığınızda görünen ekranı görüntüler.

DİKKAT

COLD KOMUTU KULLANILDIĞINDA BİLGİSAYARIN HAFIZASINDAKİ TÜM PROGRAMLAR TEMİZLENİR. İSTERSENİZ, OLD KOMUTU KULLANARAK GERİ ÇAĞIRABİLİRSİNİZ (Bkz. Bölüm 2.15). YENİ BİR PROGRAMIN HERHANGİ BİR PARÇASI GİRİLDİYSE, ÖNCEKİ PROGRAMI GERİ YÜKLEMENİN BİR YOLU YOKTUR.

ÖRNEK: COMMODORE 64'ü SIMON BASIC'in başına sıfırlamak için.

KOMUT: `COLD` **<RETURN>**

SONUÇ: İlk SIMON BASIC ekranı görüntülenir.

2.14 PROGRAM GÜVENLİK YARDIMCILARI

2.14.1 GİRİŞ

SIMON BASIC, yetkisiz kişilerin incelemesini önlemek için belirli program kodu satırlarını gizlemek üzere kullanılabilen iki komut sunar. DISAPA komutu, hangi kod satırlarını gizlemek istediğinizi belirtir. SECURE komutu, bu satırlardaki kodu boşaltır. Bu komutlar, parolaları, seri numaralarını vb. gizlemek için kullanışlıdır.

DİKKAT

GİZLİ SATIRLARI TEKRAR YAZMAK DIŞINDA BU KOMUTLARI GERİ ALMANIN BİR YOLU YOKTUR. BU NEDENLE, KULLANILMADAN ÖNCE, KENDİ KULLANIMINIZ İÇİN PROGRAMIN GÜVENLİĞİ OLMAYAN BİR KOPYASINI SAKLAMANIZ AKILLICADIR.

2.14.2 DISAPA

FORMAT: DISAPA:

AMAÇ: Bir program satırındaki kodun gizleneceğini belirtmek.

DISAPA komutu, bir program satırındaki ilk komut olarak kullanılır ve bu satırdaki kodun gizleneceğini belirtir. Daha sonra kodu gizlemek için SECURE komutu (aşağıdaki bölüme bakın) kullanılır. DISAPA komutu, kodun göründüğü her satırda kodun önüne otomatik olarak üç iki nokta üst üste (:) ekler.

Bu karakterler için satırda boşluk bırakılması gerektiğini unutmayın; yani gizlenecek bir satırın maksimum uzunluğu (DISAPA ve iki nokta üst üste hariç) 30 karakterdir.

ÖRNEK: Aşağıdaki programın 10 ila 40. satırlarındaki kodun gizleneceğini belirtmek için:

```
10 PRINT "SELAM"
20 PRINT SA
30 SA = SA + 1
40 GOTO 10
```

GİR:

```
10 DISAPA: PRINT "SELAM"
40 DISAPA: GOTO 10
```

KOMUT: `LIST` **<RETURN>**

EKRAN:

```
10 DISAPA ::::: PRINT "SELAM"
20 PRINT SA
30 SA = SA + 1
40 DISAPA :::::GOTO 10
```

SONUÇ: SECURE komutunu kullandığınızda (aşağıdaki bölüme bakın) DISAPA içeren program satırları gizlenecektir.

2.14.3 SECURE

FORMAT: SECURE 0

AMAÇ: DISAPA komutuyla başlayan tüm program satırlarını gizlemek.

SECURE komutu, DISAPA'yı içeren tüm program satırlarında (önceki bölüme bakın) kodun o satırın ilk komutu olarak listelenmesini engeller. Kod normal şekilde yürütülür.

ÖRNEK: Önceki bölümdeki örnek programda 10 ve 40. satırları gizlemek için:

KOMUT:
```
SECURE 0 <RETURN>
```

YAZ:
```
LIST <RETURN>
```

EKRAN:
```
10
20 PRINT SA
30 SA = SA + 1
40
```

SONUÇ: Program listelendiğinde, satır numaraları görüntülenmesine ve programın normal çalışmasına rağmen 10. ve 40. satırlarda herhangi bir kod bulunmadığı görülmektedir.

2.15 OLD

FORMAT: OLD

AMAÇ: NEW komutunu tersine çevirmek.

OLD, NEW komutuyla bellekten silinmiş gibi görünen bir programın geri çağrılıp tekrar çalıştırılmasını sağlar. Komut herhangi bir parametre gerektirmez. (Daha teknik bir ifadeyle, OLD komutu sıfır sayfa işaretçilerini BASIC'in başlangıcına ve sonuna sıfırlar.)

ÖRNEK: Aşağıdaki programı NEW yapmak ve sonra geri çağırmak için:

```
10 REM "*** OLD KOMUTU ***"
20 A$ = "COMMODORE 64"
30 FOR C = 1 TO LEN(A$)
40 PRINT"■                ",LEFT$(A$,C)
50 FOR X = 1 TO 100 :NEXT X,C
```

KOMUT: NEW <RETURN>

YAZ: LIST <RETURN>

EKRAN: READY.

KOMUT: OLD <RETURN>

YAZ: LIST <RETURN>

EKRAN:
```
10 REM "*** OLD KOMUTU ***"
20 A$ = "COMMODORE 64"
30 FOR C = 1 TO LEN(A$)
40 PRINT"S               ",LEFT$(A$,C)
50 FOR X = 1 TO 100 :NEXT X,C
```

ÜÇÜNCÜ BÖLÜM
GİRİŞ DOĞRULAMA VE METİN İŞLEME

3.1 GİRİŞ

Üçüncü Bölüm, karakter dizisi işleme, ekran biçimlendirme ve giriş doğrulama ile ilgili SIMON BASIC komutlarını içerir.

INSERT komutu, bir dizeyi başka bir dizenin içine ekleyerek daha büyük bir karakter dizisi oluşturmanızı sağlar. INST komutu, bir karakter dizisinin, içindeki belirli bir konumdan başka bir dizeyle üzerine yazılmasını sağlar. PLACE komutu, bir dize içindeki bir karakter grubunun konumunu belirlemenizi sağlar. DUP komutu, daha küçük bir karakter dizisini belirli sayıda çoğaltarak daha büyük bir karakter dizisi oluşturmanızı sağlar.

LIN komutu, imlecin bulunduğu satırın numarasını döndürür. CENTRE, metni bir ekran satırında ortalamanızı sağlar. PRINT AT komutu, metnin ekranda nereye yazdırılacağını belirtmenizi sağlar. USE, sayısal veri sütunlarını hizalamanızı sağlar.

FETCH komutu, kullanıcı girişi için parametreler ayarlamanıza olanak tanır. INKEY, hangi işlev tuşuna basıldığını kontrol etmenizi sağlar. ON KEY komutu, yazılanlara bağlı olarak programın belirli bir noktaya dallanmasını sağlar. DISABLE bu komutu sonlandırırken, RESUME komutu yeniden etkinleştirilmesini sağlar.

Standart COMMODORE 64 BASIC karakter dizisi komutlarıyla birlikte kullanıldığında, bu özellikler size metin dizeleri üzerinde tam manipülatif kontrol sağlar.

Bu bölümdeki komutların doğrudan modda veya bir programın parçası olarak kullanılabileceğini unutmayın.

3.2 KARAKTER DİZESİ İŞLEME

3.2.1 INSERT

FORMAT: INSERT ("alt dize","ana dize",p)

AMAÇ: Bir karakter dizesini diğerine eklemek.

INSERT, bir karakter dizesinin ortasına bir karakter grubunun yerleştirilmesini sağlayarak daha uzun bir dize oluşturur. P parametresi, alt dizenin ana dizede hangi konumdan sonra eklendiğini belirtir. Alt dize ve ana dize, tırnak işaretleri veya dize değişkenleri (örneğin "aaaaa" veya a$) arasına alınmış herhangi bir ifade olabilir. Yeni dizenin maksimum uzunluğu 255 karakterdir.

INSERT komutu ayrıca iki karakter dizisini 'true/false1' mantığını kullanarak karşılaştırmak için de kullanılabilir, yani ifade doğruysa -1 değeri, yanlışsa 0 değeri döndürülen bir mantık ifadesinde karşılaştırılır.

Bu komut yanlış kullanıldığında iki olası hata oluşabilir:

? INSERT PARAMETER TOO LARGE

Bu durum, ana dize içinde ekleme noktası olarak belirtilen konumun dize uzunluğundan daha büyük bir değer olması durumunda meydana gelir.

? CREATED STRING TOO LONG

Bu hata mesajı, INSERT komutuyla oluşturduğunuz dizenin 255 karakterden, yani BASIC'in destekleyebileceğinden büyük olması durumunda görüntülenir.

ÖRNEK: "O IYI DEDI" karakter dizisine "GUNLER" kelimesini eklemek için:

GİR:
```
100 PRINT INSERT ("GUNLER ","O IYI DEDI ",6)
```

YAZ: `RUN` **<RETURN>**

EKRAN:
```
O IYI GUNLER DEDI
```

SONUÇ: "O IYI DEDI" ana dizesinin içine yedinci karakter konumundan başlayarak "GUNLER" alt dizesi eklendi.

ÖRNEK: Daha uzun bir dize değişkeni oluşturmak için:

GİR:
```
100 B$ = "GUNLER "
105 A$ = INSERT (B$,"O IYI DEDI",6)
110 PRINT A$
```

YAZ: `RUN <RETURN>`

EKRAN:
```
O IYI GUNLER DEDI
```

YAZ: `DUMP <RETURN>`

EKRAN:
```
B$ = "GUNLER "
A$ = "O IYI GUNLER DEDI"
```

ÖRNEK: İki karakter dizisini karşılaştırmak için:

GİR:
```
100 A = (INSERT("GUNLER ","O IYI DEDI",6
) = " O IYI GUNLER DEDI")
110 PRINT A
```

YAZ: `RUN <RETURN>`

EKRAN:
```
-1
```

SONUÇ: İki dize aynı olduğundan, yani koşul doğru olduğundan, -1 değeri döndürülür. Koşul yanlış olsaydı, sıfır değeri döndürülürdü.

3.2.2 INST

FORMAT: INST ("alt dize","ana dize",p)

AMAÇ: Belirtilen konumdan başlayan bir dizenin üzerine yazmak.

INST, bir karakter dizisini, belirtilen konumdan başlayarak ana dizenin üzerine yazan başka bir dizeyle değiştirir. Alt dize veya ana dize, karakter dizisi değişkenleri (yani "aaaaa" veya XX$) oldukları sürece herhangi bir ifade olabilir. P değeri, alt dizenin ana dizenin üzerine yazılacağı SONRAKİ konumu belirtir.

Bu komutla oluşabilecek olası bir hata mesajı şudur:

? CREATED STRING TOO LONG

Bu durum, oluşturduğunuz yeni dizenin 255 karakterden uzun olması durumunda gerçekleşir.

ÖRNEK: "O İYİ İDİ" cümlesindeki "İYİ" kelimesini "DAHA İYİ" ile değiştirmek için:

GİR:
```
5 A$ = "O IYI IDI"
10 A$ = INST("DAHA IYI",A$,2)
20 PRINT A$
```

YAZ: **RUN <RETURN>**

EKRAN:
```
O DAHA IYI
```

KOMUT: **DUMP <RETURN>**

EKRAN:
```
A$ = "O DAHA IYI"
```

3.2.3 PLACE

FORMAT: PRINT PLACE ("alt dize","ana dize")

AMAÇ: Ana dizenin içindeki alt dizenin konumunu belirlemek.

PLACE, bir karakter dizisinde belirtilen bir karakter grubunu (alt dize) arar. Grup bulunursa, grubun ilk karakterinin konumu döndürülür. Eşleşme bulunamazsa sıfır değeri döndürülür. Alt dizenin uzunluğu, aranan ana dizenin uzunluğundan her zaman daha kısa olmalıdır. Bu komut, iki sayısal değişkeni karşılaştırmak için de kullanılabilir.

ÖRNEK: "DAHA IYI" alt dizisinin "O DAHA IYI" ana dizisi içindeki konumunu belirlemek için:

GİR:
```
10 A$ = INST("DAHA IYI","O IYI IDI",2)
20 PRINT PLACE ("DAHA IYI",A$)
```

YAZ: **RUN <RETURN>**

EKRAN:
```
3
```

ÖRNEK: Basit bir Türkçe Dil testi:

GİR:
```
10 PRINT"ASAGIDAKI CUMLEDE";
15 PRINT"ZARFIN ILK KARAKTERININ KONUMUN
U GIRIN":PRINT
20 PAUSE 1
30 A$ = "DISARIDA ONA BAGIRARAK O CAGIRD
I"
40 B = PLACE("BAGIRARAK",A$):B$ = "BAGIR
ARAK":PRINT A$
50 INPUT A
60 IF A = B THEN 80
70 PRINT"YANLIS":PRINT"DOGRU CEVAP" B
75 PRINT"ZARF ";B$:END
80 PRINT "IYI YAPTIN":END
```

YAZ: `RUN` <RETURN>

EKRAN:
```
ASAGIDAKI CUMLEDE
ZARFIN ILK KARAKTERININ KONUMUNU GIRIN
DISARIDA ONA BAGIRARAK O CAGIRDI
```

YAZ: `8` <RETURN>

EKRAN:
```
YANLIS
DOGRU CEVAP 14'TUR.
ZARF BAGIRARAK
```

3.2.4 DUP

FORMAT: DUP ("dize",n)

AMAÇ: Bir karakter dizesini n kez kopyalamak.

DUP, bir dizenin katlarından yeni bir karakter dizesi üretilmesini sağlar. n, eski dizenin kaç kez yeniden üretildiğini gösterir.

Oluşturduğunuz yeni dizenin 255 karakterden uzun olması durumunda aşağıdaki hata mesajının görüntüleneceğini unutmayın:

? CREATED STRING TOO LONG

ÖRNEK: Bir karakter dizesini üç kez çoğaltmak ve ardından başka bir dize eklemek için:

GİR:
```
10 A$ = DUP ("MERHABA-",3)
20 B$ = " BURADA NELER OLUYOR?"
30 C$ = A$ + B$:PRINTC$
```

YAZ: `RUN` <RETURN>

EKRAN:
```
MERHABA-MERHABA-MERHABA- BURADA NELER OL
UYOR?
```

3.2.5 CENTRE

FORMAT: CENTRE "karekter dizesi"

AMAÇ: Bir karakter dizisini ekran satırında ortalamak.

CENTRE, metnin ekran satırının ortasında görüntülenmesini sağlar. Bu komutu kullanmak için metnin uzunluğunu bilmeniz gerekmez.

ÖRNEK: "COMMODORE 64" karakter dizisini ortalamak için:

KOMUT: `CENTRE ''COMMODORE 64''` **<RETURN>**

EKRAN:

```
                    COMMODORE 64
```

3.2.6 AT

FORMAT: PRINT AT (c,r) "karakter dizesi"

Veya: PRINT "1. dize"AT(c,r)"2. dize"

AMAÇ: Belirtilen bir ekran konumunda bir karakter dizisini yazdırmak için.

AT komutu, bir karakter dizisinin yazdırılmaya başlanacağı ekran konumunu belirtmenizi sağlar. Bu, metni konumlandırmak için imleç kontrol karakterlerinin kullanılmasının yerini alır. c ve r parametreleri, parametreyi takip eden karakter dizisinin ekranda başlamasını istediğiniz konumun sütun ve satır koordinatlarını tanımlar. Tek bir ifadede birden fazla AT komutu birleştirilebilir.

ÖRNEK: "COMMODORE 64" karakter dizesini 13. sütun, 8. satıra yerleştirmek için:

KOMUT: `PRINT AT(13,8)''COMMODORE 64''` **<RETURN>**

EKRAN: Şekil 3-1'de görüldüğü gibi.

ŞEKİL 3-1 TEK BİR 'AT' KOMUTU

ÖRNEK: 13. sütun, 8. satırdan başlayarak "CBM 64" karakter dizisini ve üç satır aşağıda ve iki karakter sağda "SIMON BASIC" dizisini yazdırmak için:

KOMUT: `PRINT AT(13,8)"CBM 64"AT(15,11)"SIMON BASIC"` **<RETURN >**

EKRAN: Şekil 3-1'de görüldüğü gibi.

ŞEKİL 3-1 BİLEŞİK BİR 'AT' KOMUTU

3.2.7 USE

FORMAT: USE "###.#####",vs:PRINT

Veya: USE "##yazı.###yazı",vs:PRINT

AMAÇ: Sayısal verileri biçimlendirmek için.

USE komutu, sayı listelerini biçimlendirmenize, yani ondalık noktalarını hizalamanıza olanak tanır. Ondalık noktasının her iki tarafındaki kare işareti (#) miktarı, COMMODORE 64'e dizedeki ilgili rakam sayısını bu konuma göre görüntülemesini söyler. İsterseniz, kare işaretleri arasına metin de ekleyebilirsiniz. vs parametresi, USE komutunu kullanmak istediğiniz sayının dize gösterimidir. USE komutu satır başı gerektirmediğinden, PRINT komutunun dizeyi takip etmesi gerektiğini unutmayın.

ÖRNEK: Rastgele oluşturulmuş fiyatların tablo halindeki listesini yazdırmak için:

GİR:

```
10 REM''*** USE ORNEGI ***
20 A$ = STR$(RND(1) * 199)
30 USE ''TL####,##KR'',A$:PRINT
40 GET A$:IF A$ = "" THEN 40
50 GOTO 20
```

YAZ: `RUN` **<RETURN>**

EYLEM: Bir tuşa basın.

EKRAN:

```
TL126,45KR
```

EYLEM: Bir tuşa basın.

EKRAN:

```
TL35,36KR
```

SONUÇ: Bir tuşa her bastığınızda bir değer görüntülenir. Listedeki tüm rakamların ondalık basamakları, her ekran satırında aynı konumda görünür. Değerler rastgele üretildiğinden, yukarıda gösterilen rakamların yalnızca örnek olduğunu unutmayın.

3.3 GİRİŞ DOĞRULAMA KOMUTLARI

3.3.1 FETCH

FORMAT: FETCH "kontrol karakteri",I,belirlenen dize

AMAÇ: Kullanıcı girdisi için karakter türünü ve sayısını sınırlamak.

FETCH, klavyeden neyin kabul edileceğini kontrol etmenizi sağlar. Tırnak işaretleri içindeki kontrol karakteri, izin verilen karakter türlerini belirler. Geçerli karakter türleri ve ilişkili kontrol karakteri aşağıda gösterilmiştir:

KONTROL KARAKTER	GEÇERLİ KARAKTERLER
CLR/HOME	Sadece değiştirilmemiş alfabetik karakterler.
CURSOR DOWN	Sadece sayısal karakterler.
CURSOR RIGHT	Alfanümerik ve değiştirilmiş karakterler.

FETCH komutundaki I parametresi, kullanıcının girebileceği maksimum karakter sayısını belirten bir sayıdır. Komuttaki üçüncü parametre, girdinin yerleştirileceği dize değişkenini belirtir.

ÖRNEK: Kullanıcı girişini en fazla sekiz değiştirilmemiş alfabetik karakterle sınırlamak ve bu girişi A$ dize değişkenine yerleştirmek için:

GİR:
```
10 PRINT:PRINT"SENIN ISMIN NEDIR?"
20 FETCH "S",8,A$
30 PRINT"MERHABA "A$
```

YAZ: **RUN <RETURN>**

EKRAN:
```
SENIN ADIN NEDIR?
```

EYLEM: **<SHIFT>** tuşunu basılı tutun ve herhangi bir harfe basın.

SONUÇ: Hiçbir şey olmuyor.

EYLEM: Bir sayısal tuşa basın.

SONUÇ: Yine hiçbir şey olmuyor.

YAZ: **RECEP <RETURN>**

EKRAN:
```
MERHABA RECEP
```

KOMUT: **DUMP <RETURN>**

EKRAN:
```
A$ = "RECEP"
```

SONUÇ: A$'a giriş olarak yalnızca sekiz veya daha az değiştirilmemiş alfabetik karakterden oluşan bir dize kabul edilir.

3.3.2 INKEY

FORMAT: INKEY

AMAÇ: Bir fonksiyon tuşu girişini test etmek.

INKEY, hangi fonksiyon tuşuna basıldığını belirlemenizi sağlar. INKEY, basılan fonksiyon tuşunun numarasını (1'den 8'e kadar) gösterir. Bu komut, fonksiyon tuşlarının belirli seçenekleri veya işlemleri seçmek için kullanılabildiği menü odaklı programlarda özellikle kullanışlıdır.

ÖRNEK: F1 ve F2 fonksiyon tuşlarını test etmek için:

GİR:
```
10 A = INKEY
20 ON A GOSUB 1000,2000
30 GOTO 10
1000 PRINT "F1'E BASTINIZ":RETURN
2000 PRINT "F2'YE BASTINIZ":RETURN
```

YAZ: `RUN` **<RETURN>**

EYLEM: **<F1>** fonksiyon tuşuna basın.

EKRAN:
```
F1'E BASTINIZ
```

EYLEM: **<SHIFT>** tuşunu basılı tutun ve **<F1>** fonksiyon tuşuna basın.

EKRAN:
```
F2'YE BASTINIZ
```

3.3.3 ON KEY

FORMAT: ON KEY "karakter(ler)",:GOTO satır numarası

AMAÇ: Bir programda belirli bir noktaya dallanmak.

ON KEY komutu, COMMODORE 64'ün komutta tanımlanan karakterlerden birinin girişi için klavyeyi taramasına neden olur. Belirtilmeyen herhangi bir tuş yok sayılır. Geçerli bir karakter alındığında, program yürütme işlemi GOTO ile belirtilen satırdan devam eder. Ayrılmış değişken ST, basılan tuşun CHR$ değerini tutar. (CHR$ kodlarının tam listesi COMMODORE 64 Kullanım Kılavuzunuzda bulunabilir.) Bu komut, özellikle menü tabanlı programlarda kullanışlıdır.

NOT

ON KEY komutu çalıştırıldığında, COMMODORE 64, belirtilen aralıktaki bir karakter girilse bile klavyeyi taramaya devam eder. Bu nedenle, ON KEY kapatmak için DISABLE komutunu (aşağıdaki bölüme bakın) kullanmanız gerekir.

ÖRNEK: Geçerli giriş karakterlerinin aralığını tanımlamak için:

GİR:
```
10 PRINT"□BIR TUSA BASIN (SON ICIN S)"
20 B$ = "DGHNVMLPOS"
30 ON KEY B$,: GOTO 50
40 GOTO 20
```

SONUÇ: Programın bu bölümü çalıştırıldığında, tanımlanan aralıktaki karakterlerden biri girilene kadar program durur.

3.3.4 DISABLE

FORMAT: DISABLE

AMAÇ: ON KEY komutunu sonlandırmak.

DISABLE, ON KEY komutuyla oluşturulan klavye taramasının (önceki bölüme bakın) kapatılmasına neden olur. ON KEY komutu kullanılmışsa, bu komut HER ZAMAN kullanılmalıdır. Aksi takdirde, 'yinelemeli atlamalar' meydana gelir; yani, belirtilen karakterlerden biri her yazıldığında program her zaman ON KEY tarafından belirtilen satıra döner.

ÖRNEK: ON KEY komutunu devre dışı bırakmak için:

GİR:

```
10 PRINT"␣BIR TUSA BASIN (SON ICIN S)"
20 B$ = "DGHNVMLPOS"
30 ON KEY B$,: GOTO 50
40 GOTO 20
50 DISABLE
```

SONUÇ: Programın bu bölümü çalıştırıldığında geçerli bir karakter girildikten sonra ON KEY komutu kapatılır.

3.3.5 RESUME

FORMAT: RESUME

AMAÇ: Önceki ON KEY komutunu yeniden devreye sokmak.

RESUME komutu, tanımlanan son ON KEY komutunun tekrar açılmasını sağlar. Bu, ON KEY tarafından belirtilen aralıktaki karakterlerden biri girilene kadar programın tekrar durmasına neden olur.

ÖRNEK: Önceki bölümdeki programı genişleterek, ON KEY komutunu tekrar açmak için:

GİR:

```
10 PRINT "BIR TUSA BASIN (SON ICIN S)"
20 B$ = "DGHNVMLPOS"
30 ON KEY B$,: GOTO 50
40 GOTO 30
50 DISABLE
60 A$ = CHR$(ST): X = PLACE(A$,B$)
70 ON X GOTO 80,90,100,110,120,130,140,1
50,160,170
80 PRINT "D IDI": RESUME
90 PRINT "G IDI": RESUME
100 PRINT "H IDI": RESUME
110 PRINT "N IDI": RESUME
120 PRINT "V IDI": RESUME
130 PRINT "M IDI": RESUME
140 PRINT "L IDI": RESUME
150 PRINT "P IDI": RESUME
160 PRINT "O IDI": RESUME
170 PRINT "S IDI": END
```

YAZ: `RUN` **<RETURN>**

YAZ: `V` **<RETURN>**

EKRAN:

```
V IDI
```

YAZ: `X` **<RETURN>**

EKRAN: Hiçbir şey görüntülenmiyor.

YAZ: `E` **<RETURN>**

EKRAN:

```
E IDI

READY.
```

SONUÇ: ON KEY komutunda tanımlanan aralıktaki bir karakter, o karakter mesajının görüntülenmesine neden olur. Diğer tüm karakterler yok sayılır.

DÖRDÜNCÜ BÖLÜM
EKSTRA SAYISAL YARDIMLAR

4.1 GİRİŞ

Bu bölüm, sayısal verileri işlerken size yardımcı olacak çeşitli komutlar içerir. MOD ve DIV komutları, pozitif sayılarda tam sayı bölme işleminin gerçekleştirilmesini sağlar. Tüm sonuçlar yuvarlanmış olarak döndürülür. FRAC, bir sayının kesirli kısmını çıkarmanızı sağlar. Ayrıca, bu bölümde onaltılık veya ikili sayıları ondalık sayılara dönüştürme komutları da yer almaktadır. Ek bir Boole operatörü olan exclusive or (EXOR), bu bölümdeki komutları tamamlar.

Bu bölümdeki komutların doğrudan modda veya bir programın parçası olarak kullanılabileceğini unutmayın.

4.2 EK ARİTMETİK OPERATÖRLER

4.2.1 MOD

FORMAT: MOD(x,y)

AMAÇ: Bir tam sayının başka bir tam sayıya bölümünden kalanı döndürmek.

MOD komutu, bir tam sayının (yani tam sayının) başka bir tam sayıya bölünmesinden kalanı görüntüler. MOD komutu doğrudan veya bir program içinde kullanılabilir.

ÖRNEK: 15'i 4'e bölüp kalanı bulmak için:

YAZ:
```
PRINT MOD(15,4) <RETURN>
```

EKRAN:
```
3
```

4.2.2 DIV

FORMAT: DIV(x,y)

AMAÇ: Y ile çarpıldığında x'e eşit veya x'ten küçük olan en büyük tam sayıyı döndürmek için.

DIV komutu, bir kayan noktalı sayıyı başka bir kayan noktalı sayıya bölmenizi ve sonucu tam sayı biçiminde üretmenizi sağlar; yani sonucun kesirli kısmı göz ardı edilir.

ÖRNEK: 10'u 3'e bölüp sonucu tam sayı formunda elde etmek için:

GİR:
```
10 A = DIV(10,3)
20 PRINT A
```

YAZ: **RUN <RETURN>**

EKRAN:
```
3
```

4.2.3 FRAC

FORMAT: FRAC(n)

AMAÇ: Bir sayının kesirli kısmını döndürmek için.

FRAC, kesirli sayının, yani tam sayı olmayan sayının ondalık noktadan sonra gelen kısmını, en fazla dokuz ondalık basamağa kadar çıkarmanıza olanak tanır.

ÖRNEK: 22'yi 6'ya bölüp sonucun kesirli kısmını elde etmek için:

KOMUT: **PRINT FRAC(22/6) <RETURN>**

EKRAN:
```
.666666667
```

ÖRNEK: π'nin kesirli kısmını döndürmek için.

GİR:
```
10 PRINT π
20 PRINT FRAC(π)
```

YAZ: **RUN <RETURN>**

EKRAN:
```
3.14159265
.14159265
```

4.3 SAYISAL DÖNÜŞÜM

4.3.1 % - İKİLİ SAYIDAN ONDALIK SAYIYA DÖNÜŞTÜRME

FORMAT: PRINT %ikili sayı

AMAÇ: İkili sayı sistemini ondalık sayı sistemine dönüştürmek.

% komutu ikili sayıyı ondalık eşdeğerine dönüştürür.

Komutta argüman olarak ikili olmayan bir sayı kullanılırsa şu mesaj görüntülenir:

? NOT BINARY CHARACTER

görüntülenir.

ÖRNEK: 10110101 ikili sayısını ondalık forma dönüştürmek için:

KOMUT: `PRINT %10110101` **<RETURN>**

EKRAN: `181`

4.3.2 $ - ONALTILI SAYIDAN ONDALIKLI SAYIYA DÖNÜŞTÜRME

FORMAT: PRINT $onaltılık sayı

AMAÇ: Onaltılık sayı sistemini ondalık sayı sistemine dönüştürmek

$ komutu onaltılık bir sayıyı ondalık eşdeğerine dönüştürür.

Komutta argüman olarak onaltılık olmayan bir sayı kullanılırsa şu mesaj görüntülenir:

? NOT HEX CHARACTER

görüntülenir.

ÖRNEK: EB38 onaltılık sayısını ondalık forma dönüştürmek için:

KOMUT: `PRINT $EB38` **<RETURN>**

EKRAN: `60216`

4.3.3 DÖNÜŞTÜRME KOMUTLARININ BİRLEŞTİRİLMESİ

Yukarıdaki iki komut birlikte kullanılabilir.

ÖRNEK: İkili ve onaltılık sayıları toplayıp sonucu ondalık biçimde döndürmek:

KOMUT:
```
PRINT %10110101 + $EB38 <RETURN>
```

EKRAN:
```
60397
```

4.4 EXOR

FORMAT: EXOR(n,n1)

AMAÇ: İki sayı arasında veya özel bir işlem gerçekleştirmek

EXOR komutu, iki sayı arasında özel bir VEYA işlemi gerçekleştirmenizi sağlar. Komut önce her iki sayıyı da ikili forma dönüştürür. Ardından bu ikili sayıları bit bit karşılaştırır. Her iki bit de aynıysa, karşılık gelen sonuç biti sıfırlanır, yani 0 olur. Bitler farklıysa, karşılık gelen sonuç biti ayarlanır, yani 1 olur.

ÖRNEK: Özel veya 87 ve 45'e:

KOMUT:
```
PRINT EXOR(87,45)
```

EKRAN:
```
122
```

Bu cevaba ulaşmak için kullanılan yöntem aşağıda gösterilmiştir:

Birinci numara	= 87 = 01010111
İkinci numara	= 45 = 00101101
Sonuç	= 01111010 = 122

ÖRNEK: Ekrana karakterleri ters sırada yazdırmak için:

GİR:
```
5 PRINT"␣"
7 FOR C = 1 TO 10
8 PRINT "SIMON BASIC":PRINT:NEXT
10 FORX = 0 TO 999
20 A = PEEK (1024 + X)
30 IF A = 32 THEN 60
40 K = EXOR (A.128)
50 POKE 1024 + X,K
60 NEXT :GOTO 10
```

YAZ:
```
RUN <RETURN>
```

SONUÇ: Ekrandaki her karakter ters alana dönüştürülüyor ve ardından tekrar normale dönüyor.

BEŞİNCİ BÖLÜM
DİSKET KOMUTLARI

5.1 GİRİŞ

SIMONS' BASIC, iki basitleştirilmiş disk işleme komutu içerir. DISK komutu, bir disk sürücüsüne kanal açarken mantıksal dosya numarası, aygıt numarası ve ikincil adres belirtme ihtiyacını ortadan kaldırır. Komut ayrıca, belirtilen işlem tamamlandığında kanalı otomatik olarak kapatır. DIR komutu, BASIC kodu LOAD "$",8'in yerini alarak, tek bir komutla disket dizininin bir kısmını veya tamamını listelemenizi sağlar.

5.2 DISK

FORMAT: DISK,"işlem"

AMAÇ: Disket kanalını açmak ve işlem tamamlandığında kapatmak.

DISK aşağıdaki BASIC kodunu değiştirir:

OPEN mantıksal dosya, aygıt numarası, ikincil adresi: PRINT# mantıksal dosya

Bu komut, disket sürücüsüne bir kanal açar ve belirtilen işlem tamamlandığında bu kanalı otomatik olarak kapatır.

ÖRNEK: Yeni bir diskete "TEST" başlığını yazarak biçimlendirin:

EYLEM: Disket sürücüsüne yeni bir disket yerleştirin.

KOMUT: `DISK ""N0:TEST,01"` **<RETURN>**

SONUÇ: Birkaç dakika sonra, yeni disket "TEST" başlığıyla biçimlendirilir ve disket kanalının kapatıldığını belirtmek için sürücü ışığı söner.

ÖRNEK: Disketten bir programı kazımak:

EYLEM: Aşağıdaki kısa programı yazın ve ardından "TURUNCU" adıyla diskete kaydedin:

```
10 REM"*** DISK'E ORNEK ***
20 REM"*** BIR PROGRAM SILIN ***
```

KOMUT: `DISK "S0:TURUNCU"` **<RETURN>**

SONUÇ: "TURUNCU" programı disketten silindi ve disket kanalı kapatıldı.

5.3 DIR

FORMAT: DIR "$

Veya: DIR"$:karekter dizesi*

Veya: DIR"$:?karekter dizesi

Veya: DIR"$:?karekter dizesi*

AMAÇ: Disket dizininin bir kısmını veya tamamını listelemek.

DIR komutu, BASIC kodu olan LOAD "$",8'in yerini alır. Bu komut, disket dizininin bir kısmını veya tamamını görüntülemenizi sağlar. Yalnızca adları belirli bir karakter veya karakter dizisiyle başlayan dosyaları görüntülemek için, bu karakteri veya karakter dizisini ardından bir yıldız işareti (*) girin. İsterseniz, dosya adında belirli bir karakter veya karakter dizisinin belirli bir konumda bulunduğu dosyaları görüntülemek için, baştaki karakterleri soru işaretleriyle (?) değiştirin.

ÖRNEK: Tüm dizin listesini görüntülemek için:

KOMUT: `DIR"$` **<RETURN>**

SONUÇ: 8 numaralı cihazdaki disketin dizini görüntüleniyor.

ÖRNEK: Dosya adının üçüncü karakteri "S" olan dosyaları listelemek için:

KOMUT: `DIR"$:??S*` **<RETURN>**

SONUÇ: Ekranda, üçüncü karakteri S olan dosya adlarının dizin listesi gösterilmektedir.

ALTINCI BÖLÜM
SIMONS' BASIC İLE GRAFİKLER

6.1 GİRİŞ

Bu bölümde, SIMON BASIC'in kapsamlı grafik çizim komutları açıklanmaktadır. Bu komutlar, herhangi bir bellek konumuna erişmenize gerek kalmadan, ekranda noktalar çizmenizi, şekiller çizmenizi, metin girmenizi ve on altı renkten herhangi birini kullanarak boyama yapmanızı sağlar.

COLOUR komutu, ekranın ve onu çevreleyen çerçevenin rengini ayarlar. HIRES komutu, COMMODORE 64'ü yüksek çözünürlüklü grafik çizim moduna geçirir. Bu modda, tüm noktalar piksel piksel çizilir. MULTI, çok renkli modu başlatır. Burada, çizilen her nokta iki piksel genişliğindedir. Hem HIRES hem de MULTI komutları, grafik şekillerinizi hangi renkte çizmek istediğinizi belirtmenize olanak tanır. LOW COL komutu bu renkleri değiştirirken, HI COL komutu başlangıçta seçilen çizim renklerine geri döner.

PLOT komutu, ekrana tek noktalar çizmenizi sağlar. TEST komutu, tanımlanmış bir ekran konumunun durumunu, yani o konumda bir nokta çizilip çizilmediğini ve noktanın hangi renkte çizildiğini kontrol etmenizi sağlar. REC komutu dikdörtgenler çizmenizi, CIRCLE komutu ise dairesel şekiller çizmenizi sağlar. ARC komutu, dairesel bir şeklin çevresinin belirtilen bir bölümünü çizerken, ANGL komutu yarıçapını çizer. LINE komutu ise düz bir çizgi çizer.

PAINT komutu, bir grafik şeklini belirtilen bir renkle doldurur. BLOCK komutu, tamamen gölgelendirilmiş renk blokları görüntüler. DRAW ve ROT komutları, serbest elle bir şekil tasarlamanıza ve ardından bunu belirli bir boyutta ve dönüş açısında görüntülemenize olanak tanır.

CSET komutu, büyük/küçük harf veya büyük harf/grafik COMMODORE 64 karakter setini seçer. Bu komut ayrıca, en son gösterilen grafik ekranını geri çağırmanıza ve görüntülemenize olanak tanır. CHAR ve TEXT komutları, grafik ekranında sırasıyla tek karakterler ve karakter dizileri yazdırır. NRM komutu, grafik ekranından normal ekrana geri döner.

Altıncı bölümün ilk yarısı, ekran yapılandırmasını ve yüksek çözünürlüklü ve çok renkli grafikler arasındaki farkları ele almaktadır. On altı COMMODORE 64 rengi listelenmiş ve çizim yaparken renk seçimi nasıl yapılır gösterilmiştir. Bu bölümün ikinci yarısı, her bir grafik çizim komutunun biçimini ve kullanımını açıklamaktadır. Komutlar, bu kılavuzun On Üçüncü Bölümünde yer alanlar gibi programlar yazmak için kullanılabilecekleri sırayla listelenmiştir.

Altıncı Bölümdeki komutların, COLOUR komutu hariç, yalnızca bir programın parçası olarak kullanılabileceğini unutmayın.

6.2 EKRAN YAPILANDIRMASI

Grafik çizimi amacıyla, COMMODORE 64 ekranı bir matris veya ızgaraya bölünmüştür. Izgaradaki her nokta, tıpkı bir grafikte bir noktayı belirtir gibi x ve y koordinatlarıyla belirtilir. Örneğin, 0,0 konumu ekranın sol üst köşesini ifade eder. Izgaranın boyutu, yüksek çözünürlüklü veya çok renkli grafik modunu kullanmanıza göre değişir. Yüksek çözünürlüklü modda, ekran 320 x 200 noktalı bir matrise bölünür. Çok renkli modda ise bu matris 160 x 200 noktadır. Bu, yüksek çözünürlüklü modda çizilen her noktanın bir piksel genişliğinde, yani ekrandaki en küçük adreslenebilir nokta olduğu anlamına gelir. Çok renkli modda ise çizilen her nokta iki piksel genişliğindedir.

6.3 COMMODORE 64 RENKLERİ

Yüksek çözünürlüklü veya çok renkli modda olsun, ekranın herhangi bir 8x8 piksellik alanında yalnızca ÜÇ renk kullanılabilir. COMMODORE 64, on altı farklı çizim rengi sunar. Bu renkler ve ilgili değerleri aşağıda listelenmiştir:

0 Siyah
1 Beyaz
2 Kırmızı
3 Turkuaz
4 Mor
5 Yeşil
6 Mavi
7 Sarı
8 Turuncu
9 Kahverengi
10 Açık Kırmızı
11 Gri 1
12 Gri 2
13 Açık Yeşil
14 Açık Mavi
15 Gri 3

Grafik çizerken, şekil için kullanmak istediğiniz renk, ilgili renk numarası cinsinden belirtilir.

6.4 ÇİZİM TÜRLERİ

SIMON BASIC grafik çizme komutlarının hepsinin ortak bir özelliği vardır: Her biri bir 'çizim türü' belirtmenizi gerektirir. Bu, COMMODORE 64'e her noktanın nasıl çizileceğini söyler. Hem yüksek çözünürlüklü hem de çok renkli modlar için çizim türleri aşağıda listelenmiştir:

YÜKSEK ÇÖZÜNÜRLÜK MODU

ÇİZİM TÜRÜ	GERÇEKLEŞTİRİLEN FONKSİYON
0	Noktayı temizler.
1	Ekrana bir nokta çizer.
2	Noktanın yönünü değiştirir, yani açıksa kapatır veya kapalıysa açar.

ÇOK RENKLİ MOD

ÇİZİM TÜRÜ	GERÇEKLEŞTİRİLEN FONKSİYON
0	Noktayı temizler.
1	MULTI/LOW COL komutunun 1. renginde bir nokta çizer.
2	MULTI/LOW COL komutunun 2. renginde bir nokta çizer.
3	MULTI/LOW COL komutunun 3. renginde bir nokta çizer.
4	Noktanın rengini tersine çevirir, yani: 0 renginde çizilen bir nokta 3 rengine dönüşür 1 renginde çizilen bir nokta 2 rengine dönüşür 2 renginde çizilen bir nokta 1 rengine dönüşür 3 renginde çizilen bir nokta 0 rengine dönüşür

6.5 GRAFİK ÇİZİM KOMUTLARI

6.5.1 COLOUR

FORMAT: COLOUR bo,sc

AMAÇ: Ekran arka planını ve kenar renklerini ayarlamak.

COLOUR komutu, düşük çözünürlüklü ekranlar için ekran arka planının rengini ve hem düşük hem de yüksek çözünürlüklü ekranları çevreleyen kenarlığın rengini belirtmenizi sağlar. sc parametresi ekran arka plan rengini, bo parametresi ise kenarlık rengini ifade eder. Renkler, ilgili renk numaraları belirtilerek seçilir (bkz. Bölüm 6.3).

Düşük çözünürlüklü ekranlarda arka plan renginin, COLOUR komutu farklı bir renk değeri kullanılarak tekrar çalıştırılana kadar seçilen renkte kalacağını unutmayın. Yüksek çözünürlüklü veya çok renkli grafikler içeren bir ekranın arka plan rengi, HIRES komutu kullanılarak seçilir (bir sonraki bölüme bakın).

ÖRNEK: Ekran arka plan rengini turkuaz, kenar rengini ise mavi olarak belirtmek için:

KOMUT:
```
COLOUR 3,6 <RETURN>
```

SONUÇ: Mavi bir çerçeveyle çevrili turkuaz renginde bir ekran görüntülenir.

6.5.2 HIRES

FORMAT: HIRES pc,sb

AMAÇ: Yüksek çözünürlüklü grafik modunu başlatmak ve çizim rengini ve ekran arka plan rengini seçmek.

HIRES komutu, ekranı yüksek çözünürlüklü grafik moduna ayarlar; yani tüm noktalar piksel piksel çizilir. İlk parametre olan pc, kullanmak istediğiniz çizim renginin renk numarasıdır (bkz. Bölüm 6.3). İkinci parametre olan sb, çizimin gerçekleştiği her 8x8 piksellik karenin arka plan rengini belirtir. HIRES komutu ile ilk parametresi arasında bir boşluk olması gerektiğini unutmayın.

ÖRNEK: Grafik rengini siyah veya beyaz arka plan olarak seçmek için

GİR:
```
10 HIRES 0,1
20 GOTO 20
```

YAZ:
```
RUN <RETURN>
```

SONUÇ: Ekranda boş beyaz bir görüntü beliriyor.

EYLEM: **<RUN/STOP>** tuşuna basın.

SONUÇ: Normal ekran belirir.

6.5.3 REC

FORMAT: RECx,y,A,B,çizim türü

AMAÇ: Dikdörtgen çizmek.

REC komutu, grafik ekranında dikdörtgen bir şekil çizer. Komutun ilk parametreleri (x,y), dikdörtgenin sol üst köşesinin koordinatlarını belirtir. x, sol kenardan uzaklığı, y ise ekranın üstünden uzaklığı gösterir. A parametresi, dikdörtgenin sol üst köşesinden sağ üst köşesine olan uzaklığı, B parametresi ise sol üst köşesinden sol alt köşesine olan uzaklığı gösterir. Çizim türü, Bölüm 6.4'te açıklandığı gibidir.

ÖRNEK: Yüksek çözünürlüklü grafiklerde ekranın sol üst köşesine dikdörtgen çizmek için:

GİR:
```
10 HIRES 0,1
20 REC 0,0,40,20,1
30 GOTO 30
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Siyah bir dikdörtgen görüntüleniyor.

EYLEM: **<RUN/STOP>** tuşuna basın.

SONUÇ: Normal ekran görüntülenir.

6.5.4 MULTI

FORMAT: HIRES pc,sb: MULTI d,c2,c3

AMAÇ: Çok renkli grafik modunu başlatmak ve üç çizim rengi seçmek için.

HIRES komutundan sonra kullanıldığında MULTI, tüm çizimlerin çok renkli grafik modunda yapılmasını sağlar; yani çizilen her nokta iki piksel genişliğinde olur. MULTI'yi takip eden üç parametre, kullanmak istediğiniz çizim renklerini tanımlar. Her çizim rengi, MULTI komutundaki konumuna göre bir çizim komut dizisinde 'çizim türü' olarak seçilerek belirlenir (Bölüm 6.4'e bakınız).

ÖRNEK: Çok renkli grafik moduna girmek için, çizim renkleri olarak siyah, kırmızı ve maviyi belirtin ve ardından üç dikdörtgen çizin:

GİR:
```
10 HIRES 0,1: MULTI 0,2,6
30 REC 0,0,40,20,1
40 REC 20,20,40,20,2
50 REC 40,40,40,20,3
60 GOTO 60
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Bir siyah, bir kırmızı ve bir mavi dikdörtgen çizildi.

EYLEM: **<RUN/STOP>** tuşuna basın.

SONUÇ: Normal ekran görüntülenir.

6.5.5 NRM

FORMAT: NRM

AMAÇ: Grafik ekranından düşük çözünürlüklü ekrana geri dönmek.

NRM komutu, yüksek çözünürlüklü veya çok renkli grafik ekranını temizlemenize ve düşük çözünürlüklü ekrana geri dönmenize olanak tanır.

ÖRNEK: Önceki bölümdeki programı kullanarak, grafik ekranı beş saniye görüntülendikten sonra normal ekrana geri dönmek:

GİR:

```
10 HIRES 0,1: MULTI 0,2,6
30 REC 0,0,40,20,1
40 REC 20,20,40,20,2
50 REC 40,40,40,20,3
60 PAUSE 5
70 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Beş saniye boyunca üç dikdörtgen görüntülenir. Ardından normal ekran belirir ve üzerinde "HAZIR" yazısı ve yanıp sönen bir imleç görünür.

6.5.6 LOW COL

FORMAT: LOW COL c1,c2,c3

AMAÇ: Grafik renklerini değiştirmek için.

LOW COL komutu, HIRES veya MULTI komutlarıyla başlangıçta seçilenlerden farklı bir grafik çizim renkleri kümesi belirtmenizi sağlar.

NOT

Yüksek çözünürlüklü grafik çiziminde yalnızca iki renk kullanıldığı için (bkz. Bölüm 6.5.2), LOW COL komutundaki üçüncü rengin yüksek çözünürlükte hiçbir etkisi yoktur. Bununla birlikte, hangi grafik modu başlatılmış olursa olsun, LOW COL komutunda üç sayı kullanılmalıdır.

ÖRNEK: Yüksek çözünürlüklü grafik modunda siyah bir dikdörtgen çizmek ve çizgilerini sarı renkle gölgelendirmek için:

GİR:

```
10 HIRES 0,1
20 LOW COL 0,7,0
30 REC 20,20,60,60,1
40 PAUSE 5
50 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Beyaz ekrana siyah bir dikdörtgen çizilir. Çizim yapılan her 8x8 piksellik dikdörtgen sarı renkle boyanır. Beş saniye sonra normal ekran görüntülenir.

ÖRNEK: Orijinal çizim renklerinin her birini kullanarak çok renkli grafik modunda üç dikdörtgen çizmek, ardından bu renkleri değiştirmek ve yeni renklerin her birinde üç dikdörtgen çizmek:

GİR:

```
10 HIRES 0,1: MULTI 2,3,6: Z = 10
20 FOR X = 1 TO 3
30 REC 10,Z,30,30,X
40 Z = Z + 40: NEXT
50 LOW COL 4,5,7: Z = 10
60 FOR X = 1 TO 3
70 REC 50,Z,30,30,X
80 Z = Z + 40: NEXT
90 PAUSE 5
100 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Her biri farklı renkte altı dikdörtgen çizildi. Beş saniye sonra normal ekran görüntülendi.

6.5.7 HI COL

FORMAT: HI COL

AMAÇ: Başlangıçta seçilen çizim renklerine geri dönmek.

HI COL komutu, LOW COL (önceki bölüme bakın) kullanılmışsa, orijinal çizim renklerinizi, yani HIRES veya MULTI komutuyla başlangıçta ayarlanan renkleri geri yüklemenizi sağlar.

ÖRNEK: Farklı renklerde dokuz dikdörtgen çizmek:

GİR:

```
10 HIRES 0,1: MULTI 2,3,6: Z = 10
20 FOR Y = 10 TO 50 STEP 40
30 FOR X = 1 TO 3
40 REC Y,Z,30,30,X
50 Z=Z+40:NEXTX:Z=10:LOW COL 4,5,7:NEXTY
60 HI COL
70 FOR X = 1 TO 3
80 REC Y,Z,30,30,X
90 Z = Z + 40:NEXT
100 PAUSE 5
110 NRM
```

YAZ: **RUN <RETURN>**

SONUÇ: Orijinal grafik renklerinde üç dikdörtgen, LOW COL komutuyla atanan renklerde üç dikdörtgen ve yine orijinal grafik renklerini kullanan üç dikdörtgen daha görüntülenir. Beş saniye sonra normal ekran görüntülenir.

6.5.8 PLOT

FORMAT: PLOT x,y, çizim türü

AMAÇ: Tek bir noktayı işaretlemek.

PLOT komutu, grafik ekranında tek bir nokta çizer. x ve y parametreleri, çizilecek noktanın sırasıyla yatay ve dikey ekran koordinatlarını belirtir. Çizim türü, Bölüm 6.4'te açıklandığı gibidir.

ÖRNEK: Çok renkli grafik modunda siyah bir nokta çizmek için:

GİR:

```
10 HIRES 0,1:MULTI 0,1,2
20 PLOT 80,100,1
30 PAUSE 5
40 NRM
```

YAZ: **RUN <RETURN>**

SONUÇ: Ekranın ortasına siyah bir nokta çizildi. Beş saniye sonra normal ekran görüntülendi.

ÖRNEK: Noktalı bir eğri çizmek için:

GİR:

```
10 HIRES 0,1
20 FOR X = 0 TO 320 STEP .5
30 Y = 100 + SIN(X/30)*90
40 PLOT X,Y,1
50 NEXT
1000 GOTO 1000
```

YAZ: **RUN <RETURN>**

SONUÇ: Siyah bir sinüs dalgası çizilmiştir.

6.5.9 TEST

FORMAT: değişken = TEST (x,y)

AMAÇ: Ekranda belirtilen bir konumda bir şeyin çizilip çizilmediğini belirlemek.

TEST komutu, bir grafik ekranındaki bir konumun durumunu incelemenizi sağlar. x ve y parametreleri, test edilen noktanın ekran koordinatlarıdır. Bu noktada bir nokta çizilmişse, noktanın çizim türü döndürülür (bkz. Bölüm 6.4). Nokta yoksa 0 değeri döndürülür. Nokta, bir grafik şeklinin herhangi bir parçası olabilir.

ÖRNEK: Bir çizginin başka bir çizgiye değdiğinde sonlanmasını sağlamak için:

GİR:

```
10 REM"*** TEST'TE ORNEK ***
20 HIRES 0,1
25 FOR X = 0 TO 200
30 PLOT200,X,1:NEXT
40 FOR I = 1 TO 320
50 IF TEST(1,100) = 1 THEN 70
60 PLOT 1,100,1 :NEXT
70 PAUSE 5
80 NRM
```

YAZ: **RUN <RETURN>**

SONUÇ: Yatay çizgi dikey çizgiye değdiğinde durur. Beş saniye sonra normal ekran görüntülenir.

6.5.10 LINE

FORMAT: LINE x,y,x1,y1,çizim türü

AMAÇ: Bir çizgi çizmek.

LINE komutu, ekrandaki bir noktadan diğerine bir çizgi çizer. x ve y parametreleri, çizginin başlangıç noktasının ekran koordinatlarıdır. x1 ve y1 parametreleri ise çizginin bitiş noktasının koordinatlarıdır. Çizim türü, Bölüm 6.4'te açıklandığı gibidir.

ÖRNEK: Ekranın ortasına çapraz bir çizgi çizmek için:

GİR:
```
10 HIRES 0,1
20 LINE 0,0,320,200,1
30 PAUSE 5
40 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Ekranın sol üst köşesinden sağ alt köşesine siyah bir çizgi çizilir. Beş saniye sonra normal ekran görüntülenir.

6.5.11 CIRCLE

FORMAT: CIRCLE x,y,xr,yr,çizim türü

AMAÇ: Dairesel bir şekil çizmek.

CIRCLE komutu, grafik ekranında dairesel bir şekil çizmenizi sağlar. x ve y parametreleri, çizmek istediğiniz şeklin merkezinin ekran koordinatlarını belirtir. xr ve yr parametreleri ise sırasıyla şeklin yatay ve dikey yarıçaplarını gösterir. Bu yarıçapları değiştirerek, farklı boyutlarda daireler ve elipsler çizebilirsiniz. Çizim türü, Bölüm 6.4'te açıklandığı gibidir.

NOT

Ekran kare değil dikdörtgen olduğu için, aynı uzunluktaki x ve y yarıçapları ekranda mükemmel bir daire çizmenizi sağlamayacaktır. Bunu yüksek çözünürlük modunda yapmak için, x yarıçapı y yarıçapının 1,4 katına eşit olmalıdır. Çok renkli modda ise x yarıçapı y yarıçapının 1,6 katına eşit olmalıdır. Ancak, yuvarlak şekiller elde etmek için, daireler içeren bir Çok Renkli veya Yüksek Çözünürlüklü ekranı yazıcıya aktarmak istiyorsanız, x ve y yarıçaplarının değerleri eşit olmalıdır. Yazdırma ayrıntıları için Bölüm 7.13.2'ye bakın.

ÖRNEK: Yüksek çözünürlük modunda dairesel bir şekil çizmek için:

GİR:
```
10 HIRES 0,1
20 CIRCLE 160,100,52,40,1
30 PAUSE 5
40 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Ekranın ortasına siyah bir daire çizilir. Beş saniye sonra normal ekran görüntülenir.

ÖRNEK: Çok renkli grafik modunda elips çizmek için:

GİR:
```
10 HIRES 0,1:MULTI 2,3,4
20 CIRCLE 80,100,60,30,1
30 PAUSE 5
40 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Kırmızı bir elips çizilir. Beş saniye sonra normal ekran görüntülenir.

6.5.12 ARC

FORMAT: ARC x,y,sa,ea,i,xr,yr,çizim türü

AMAÇ: Dairesel bir şeklin yayını çizmek.

ARC komutu, dairesel bir şeklin çevresinin bir bölümünü çizmenizi sağlar. x ve y parametreleri, yayın çizildiği dairesel şeklin merkezinin ekran koordinatlarıdır. sa ve ea parametreleri sırasıyla yayın başlangıç ve bitiş açılarını tanımlar. i parametresi, çizim artışını, yani yay üzerindeki her nokta arasındaki derece cinsinden aralığı belirtir. Katı bir yay elde etmek için bu değer 1'dir. Daha büyük bir değer, yayı oluşturan noktaları birbirinden ayırır. xr ve yr parametreleri, yayın parçası olduğu dairesel şeklin sırasıyla dikey ve yatay yarıçaplarını gösterir. Çizim türü, Bölüm 6.4'te açıklandığı gibidir.

ÖRNEK: Aynı dairesel şekle sahip iki yay çizmek için:

GİR:
```
10 HIRES 0,1
20 ARC 160,100,30,150,1,40,40,1
30 ARC 160,100,210,330,1,40,40,1
40 PAUSE 5
50 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Siyah 'parantez' şeklinde iki adet parantez çizilir. Beş saniye sonra normal ekran görüntülenir.

6.5.13 ANGL

FORMAT: ANGL x,y,açı,xr,yr,çizim türü

AMAÇ: Bir dairenin yarıçapını çizmek.

ANGL komutu, çevresini göstermeden bir dairenin yarıçapını çizmenizi sağlar. x ve y parametreleri, dairenin merkezinin ekran koordinatlarıdır; 'açı' ise yarıçapın dik doğruya göre çizildiği açıdır (derece cinsinden), örneğin 45 derecelik bir açıyla çizilen bir yarıçap, saat kadranında saat 3 konumunda olur. xr ve yr parametreleri ise yarıçapın parçası olduğu dairesel şeklin sırasıyla yatay ve dikey yarıçaplarıdır. Çizim türü, Bölüm 6.4'te açıklandığı gibidir.

ÖRNEK: Bir tekerlek çizmek için:

GİR:

```
10 HIRES 0,1
20 CIRCLE 160,100,40 * 1.4,40,1
30 CIRCLE 160,100,45 * 1.4,45,1
40 FOR X = 0 TO 360 STEP 22.5
50 ANGL 160,100,X,40 * 1.4,40,1
60 NEXT
70 PAUSE 10
80 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Siyah, 'tekerlekli' bir çark çizilir. On saniye sonra normal ekran görüntülenir.

ÖRNEK: Bir yelpaze çizmek:

GİR:

```
10 HIRES 0,1
20 FOR X = 0 TO 170 STEP 5
30 ANGL 160,100,X,40,40,1
40 NEXT
50 FOR X = 170 TO 0 STEP-5
60 ANGL 160,100,X,40,40,0
70 NEXT
80 PAUSE 10
90 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Bir fan açılıp kapatılıyor. On saniye sonra normal ekran görüntüleniyor.

6.5.14 PAINT

FORMAT: PAINT x,y,çizim türü (sadece 0,1,2,3)

AMAÇ: Kapalı bir alanı renklendirmek

PAINT komutu, bir grafik şeklini, çizim türü tarafından tanımlanan renkle doldurur (Bölüm 6.4'e bakınız). Boyanacak alan tamamen kapalı olmalıdır, aksi takdirde boyama tüm ekrana yayılır. Boyanacak alan, sınırları içindeki herhangi bir noktanın x ve y koordinatları ile belirtilir. Yüksek çözünürlük modunda, aynı alan yalnızca bir kez boyanabilir. Bu durum, ekranı temizleyerek, LOW COL komutuyla çizim renklerini değiştirerek (Bölüm 6.5.6'ya bakınız), şekli yeniden çizerek ve ardından tekrar boyayarak aşılabilir. Çok renkli modda, aynı alan istediğiniz kadar farklı bir renkle boyanabilir.

ÖRNEK: Siyah bir dikdörtgen çizmek ve onu sarı renkle boyamak:

GİR:

```
10 HIRES0,1
20 REC 120,60,40,40,1
30 LOW COL 7,1,0
40 PAINT 130,70,1
50 PAUSE 5
60 NRM
```

YAZ: RUN <RETURN>

SONUÇ: Ekranın ortasına siyah bir kare çizilir ve içi sarı renkle doldurulur. Beş saniye sonra normal ekran görüntülenir.

ÖRNEK: Renkli pasta grafiği çizmek için:

GİR:

```
10 HIRES 0,1:MULTI 5,4,6
20 CIRCLE 80,100,48,78,1
30 ANGL 80,100,120,48,78,1
40 ANGL 80,100,160,48,78,1
50 ANGL 80,100,220,48,78,1
60 ANGL 80,100,330,48,78,1
70 PAINT 90,35,1
80 PAINT 60,60,3
90 PAINT 90,120,2
105 LOW COL 7,4,6
110 PAINT 80,110,1
120 PAUSE 5
130 NRM
```

YAZ: RUN <RETURN>

SONUÇ: Ekranın ortasına dört dilimli bir pasta grafiği çizilir ve her dilim farklı bir renkle boyanır. Beş saniye sonra normal ekran görüntülenir.

6.5.15 BLOCK

FORMAT: BLOCK x,y,x1,y1,çizim türü

AMAÇ: Tamamen gölgelendirilmiş bir renk bloğu çizmek.

BLOCK komutu, bir dikdörtgen çizer ve aynı anda renklendirir. Bu tek komut, REC komutuyla dikdörtgen çizmek ve PAINT komutuyla renklendirmekle aynı işlevi görür. BLOCK komutu her iki işlemi de aynı anda gerçekleştirir. Ancak, BLOCK komutuyla dikdörtgenin kenarlarının renginin, şeklin iç kısmının rengiyle aynı olduğunu unutmayın.

BLOCK komutu, farklı renklerdeki birkaç bitişik bloğu çizgilerle ayırmadan oluşturmak istediğinizde kullanışlıdır. x ve y parametreleri, görüntülemek istediğiniz renk bloğunun sol üst köşesini belirtir. x1 ve y1 parametreleri ise bloğun sağ alt köşesinin koordinatlarıdır. Çizim türü, Bölüm 6.4'te açıklandığı gibidir.

ÖRNEK: Farklı renklerde iki blok çizmek:

GİR:

```
10 HIRES 0,1: MULTI 2,6,1
20 BLOCK 10,50,50,90,1
30 BLOCK 51,50,90,90,2
40 PAUSE 5
50 NRM
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Kırmızı bir blok, mavi bir bloğun yanında görüntülenir. Beş saniye sonra normal ekran görüntülenir.

6.5.16 DRAW

FORMAT: DRAW "nnnnnnn....9",x,y,çizim türü

AMAÇ: Bir şekil tasarlamak.

DRAW komutu, bir şekil tasarlamanıza ve ardından bunu ekranda görüntülemenize olanak tanır. Şekil, kalemi kaldırmadan bir kâğıda resim çizmekle aynı şekilde tasarlanır. Ancak önemli bir fark vardır: COMMODORE 64'e kalemi hareket ettirmesini ancak iz bırakmamasını söyleyebilirsiniz. (Örnek için Bölüm 6.5.17'ye bakın.)

DRAW komutundaki x ve y parametreleri, şeklin çiziminin başladığı ekran noktasının, yani orijininin koordinatlarıdır. Tırnak işaretleri içindeki her n, COMMODORE 64'e şekli çizerken kalemin nasıl hareket ettirileceğini söyleyen bir talimattır. Herhangi bir program satırında tırnak işaretleri içine en fazla 74 talimat yerleştirilebilir. Ancak, en fazla 255'e kadar talimat dizilerini bir araya ekleyebilirsiniz. Şekle devam etmek için, önceki dizinin bittiği yerden başlayan yeni bir orijin belirtilmelidir. Her talimat ve karşılık gelen numarası aşağıda gösterilmiştir:

NUMARA	TALİMATI
0	Bir piksel sağa hareket et.
1	Bir piksel yukarı hareket et.
2	Bir piksel aşağı hareket et.
3	Bir piksel sola hareket et.
5	Bir piksel sağa hareket et ve bir nokta çiz.
6	Bir piksel yukarı hareket et ve bir nokta çiz.
7	Bir piksel aşağı hareket et ve bir nokta çiz.
8	Bir piksel sola hareket et ve bir nokta çiz.
9	Çizimi durdur.

Yukarıdaki talimatların yalnızca bir şekil tasarladığını unutmayın. Şekil, DRAW ve ROT komutlarının her ikisi de programa dahil edilene kadar (bir sonraki bölüme bakın) çizilemez; yani DRAW komutu şekli belirtmeli ve ROT komutu onu oluşturmalıdır. Çizim türü, Bölüm 6.4'te açıklandığı gibidir.

ÖRNEK: Bir zil tasarlamak için:

GİR:

```
10 A$ = "57575757878787575757575777777
77777757575757575757888888888888"
20 A$ = A$ + "88888888888888656565656565
656666666666666"
30 A$ = A$ + "56565656565686865656565"
```

SONUÇ: Programın bu bölümü çalıştırıldığında, bir zil için tasarım talimatları A$ değişkenine kaydedilir.

6.5.17 ROT

FORMAT: ROT r,s

AMAÇ: Bir şekli belirtilen yönde ve boyutta görüntülemek.

ROT komutu, DRAW komutuyla (yukarıdaki bölüme bakın) oluşturulan bir şekli, tanımlanmış bir boyutta ve belirli bir dönüş açısıyla görüntülemenizi sağlar. r parametresi, şeklin başlangıç noktasına (yani şeklin çizildiği ekran üzerindeki noktaya) göre ne kadar döndürüleceğini belirtir. Bu r değeri (0 ile 7 arasında değişir), aşağıdaki tabloda gösterildiği gibi dönüş açısını tanımlar:

DÖNÜŞ NUMARASI	DÖNME DERECELERİ
0	0
1	45
2	90
3	135
4	180
5	225
6	270
7	315

ROT komutundaki ikinci parametre, tasarladığınız şeklin görüntülenecek boyutunu tanımlar. Bu konumda "1" değeri, şeklin normal boyutta görüntüleneceğini, yani çizim komutundaki her parametrenin bir pikseli temsil ettiğini gösterir. Bu değerdeki herhangi bir artış, boyutta da karşılık gelen bir artışa neden olur.

NOT

Tasarladığınız şekil için çok büyük bir boyut belirtirseniz, görüntülendiğinde ekrandan kaybolacaktır. Bu nedenle, bu rakamın her zaman gerçekçi bir seviyede tutulduğundan emin olun.

ÖRNEK: Önceki bölümde tasarlanan şekli normal ve büyütülmüş boyutta görüntülemek için:

GİR:

```
10 A$ = "57575757878787575757575777777
77777775757575757575788888888888888"
20 A$ = A$ + "88888888888888656565656565
65666666666666"
30 A$ = A$ + "56565656565686865656565"
40 HIRES 0,1
45 FOR Y = 0 TO 7
50 FOR X = 1 TO 3
60 ROT Y,X
70 DRAW A$,160,80,1
80 PAUSE 1
90 DRAW A$,160,80,0
100 NEXT:NEXT
110 FOR X = 3 TO 1 STEP-1
115 FOR Y = 7 TO 0 STEP-1
120 ROT Y,X
130 DRAW A$,160,80,1
140 PAUSE 1:DRAW A$, 160,80,0:NEXT:NEXT
150 GOTO 45
```

YAZ: `RUN` **<RETURN**

SONUÇ: Bir çan, üç farklı boyutta ve yedi farklı dönüş açısında sergilenmiştir.

6.5.18 CSET

FORMAT: CSET n

AMAÇ: COMMODORE 64 karakter setlerinden birini seçmek veya son grafik ekranını geri çağırıp görüntülemek için.

CSET komutu, n parametresinin değerine bağlı olarak üç işlevden birini yerine getirir. CSET 0, COMMODORE 64 Büyük Harf/Grafik karakter setini seçmenizi sağlar. CSET 1, Büyük/Küçük Harf karakter setini seçmenizi sağlar. CSET 2, en son gösterilen grafik ekranını yeniden görüntüler.

NOT

Çok renkli bir grafik ekranını geri çağırırken, CSET 2'yi her zaman MULTI komutuyla (bkz. Bölüm 6.5.3) ve bu komuta başlangıçta atanmış olan aynı parametrelerle birlikte kullanmalısınız.

ÖRNEK: Bir dizeyi alternatif karakter kümeleri kullanarak yazdırmak için:

GİR:

```
10 PRINT"□"
20 PRINT AT (12,14)"SIMON BASIC"
30 CSET 1:PAUSE 1:CSET 0:PAUSE 1
40 GOTO 20
```

YAZ: RUN <RETURN>

SONUÇ: Ekranın ortasında önce büyük harflerle, sonra küçük harflerle "SIMON BASIC" karakter dizisi görüntülenir.

ÖRNEK: Önceden oluşturulmuş yüksek çözünürlüklü bir ekranı yeniden görüntülemek için:

GİR:

```
10 HIRES 0,1:MULTI 0,4,6
15 FOR I = 1 TO 20
20 A=INT(90*RND(1))+2:B=INT(90*RND(1))+2
25 C=INT(90*RND(1))+2:D=INT(60*RND(1))+2
27 P = INT(3 * RND(1)) + 1
28 IF A + C > 159 THEN 25
29 IF B + D > 199 THEN 25
30 REC A,B,C,D,1
35 PAINT A + 1,B + 1,P
37 NEXT I
40 PAUSE 2:CSET 0
50 PRINT"□YENIDEN GOSTERMEK ICIN HERHANG
   I BIR TUSA BASIN"
60 PRINT"(@) SON EKRAN"
70 GET A$: IF A$ = "" THEN 70
80 CSET 2: MULTI 0,4,6: PAUSE 2:CSET 0
90 GOTO 50
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Çok renkli bir grafik ekrana 20 renkli kare çizilir. Kısa bir gecikmenin ardından, "SON EKRANI TEKRAR GÖRÜNTÜLEMEK İÇİN HERHANGİ BİR TUŞA BASIN" mesajıyla normal ekran belirir.

EYLEM: Herhangi bir tuşa basın.

SONUÇ: Grafik ekranı yeniden görüntülenir.

6.6 GRAFİK EKRANINDA METİN YAZDIRMA

6.6.1 CHAR

FORMAT: CHAR x,y,poke kodu,çizim türü,boyut

AMAÇ: Bir karakteri grafik ekranına yazdırmak.

CHAR komutu, metni karakter karakter yüksek çözünürlüklü veya çok renkli bir grafik ekranında görüntülemenizi sağlar. x ve y parametreleri, karakterin ekrandaki konumunu belirtir. Komuttaki bir sonraki parametre, görüntülemek istediğiniz karakterin 'poke' kodudur. (Poke kodlarının tam listesi COMMODORE 64 Kullanım Kılavuzunuzda yer almaktadır.) Çizim türü, Bölüm 6.4'te açıklandığı gibidir. Bu komuttaki son parametre, karakterin yüksekliğini 1 ile 8 arasında belirtir. Bu konumda "1", karakterin normal boyutunda, yani sekiz piksel yüksekliğinde görüntüleneceğini gösterir. Bu değerdeki herhangi bir artış, karakter yüksekliğinde de karşılık gelen bir artışa neden olur; örneğin, 3 değeri karakteri 24 piksel yüksekliğinde görüntüler. Karakterlerin genişliği DEĞİŞTİRİLEMEZ.

NOT

Kullanıcı tanımlı grafik karakterler, yüksek çözünürlüklü veya çok renkli grafik ekranlarda KULLANILAMAZ.

ÖRNEK: Karakterleri normal boyutlarının iki katı olarak görüntülemek için:

GİR:

```
10 REM"*** KAREKTER ICIN ORNEK ***
20 HIRES 0,1
30 FOR J = 0 TO 15
40 FOR K = 0 TO 31
50 CHAR I*8,A,K+J*32,1,2
51 I=I+1 : IF I=40 THEN I=0 : A=A+15
60 NEXT:NEXT
70 PAUSE 5
80 NRM
```

YAZIN: RUN <RETURN>

SONUÇ: COMMODORE 64 karakter setinin tamamı normal yüksekliğinin iki katı olarak görüntülenir. Beş saniyelik bir gecikmenin ardından normal ekran görüntülenir.

6.6.2 TEXT

FORMAT: TEXT x,y,"(CTRL a) karakter dizisi ",çizim türü,s,i

Veya: TEXT x,y,"(CTRL b) karakter dizisi ",çizim türü,s,i

AMAÇ: Bir karakter dizisini grafik ekrana yazdırmak.

TEXT komutu, grafik ekranda karakter dizeleri yazdırmanıza olanak tanır. x ve y parametreleri, dizenin ilk harfinin ekran koordinatlarını belirtir. Sonraki parametre ise dizenin kendisidir. Dizenin önündeki kontrol karakteri, metnin büyük veya küçük harflerle mi görüntüleneceğini gösterir. Metni büyük harflerle görüntülemek için:

1. İlk tırnak işaretlerini yazın.

2. CTRL tuşunu basılı tutun ve 'a' tuşuna basın. (Ters alan 'A' görüntülenir.)

3. 3. Karakter dizisini girin.

4. 4. Son tırnak işaretlerini yazın.

Metni küçük harflerle görüntülemek için:

1. İlk tırnak işaretlerini yazın.

2. CTRL tuşunu basılı tutun ve 'b' tuşuna basın. (Ters alan 'B' görüntülenir.)

3. 3. Karakter dizisini girin.

4. 4. Son tırnak işaretlerini yazın.

Aynı karakter dizisinde büyük ve küçük harfleri de karıştırabilirsiniz. Bunu yapmak için, büyük harf olarak görüntülemek istediğiniz karakterlerden önce CTRL tuşunu basılı tutun ve 'a' tuşuna basın; küçük harf olarak görüntülemek istediğiniz karakterlerden önce ise CTRL tuşunu basılı tutun ve 'b' tuşuna basın.

Grafik türü, Bölüm 6.4'te açıklandığı gibidir. s parametresi, dizedeki her karakterin yüksekliğini 1 ile 8 arasında belirtir. Bu konumda "1" normal boyutlu karakterleri belirtir. Bu değerdeki herhangi bir artış, karakter boyutunda da karşılık gelen bir artışa neden olur; örneğin, karakter boyutunu 8 olarak belirtirseniz, metin normal yüksekliğinin sekiz katı olarak görüntülenir. Karakterlerin genişliği DEĞİŞTİRİLEMEZ. Son parametre i, dizedeki her karakter arasındaki piksel sayısını tanımlar. Normal aralık için bu değer 8'dir. Bu değerdeki herhangi bir artış, karakterler arasında karşılık gelen daha büyük bir boşluk oluşturur.

ÖRNEK: Yüksek çözünürlüklü bir ekranda iki karakter dizisini görüntülemek için:

GİR:

```
10 REM"*** TEXT ICIN ORNEK ***
20 HIRES 0,1
30 FOR I = 1 TO 30
40 X=INT(320*RND(1)):Y=INT(200*RND(1))
50 LINE 160,100,X,Y,1:NEXT
60 TEXT 60,20,"[B]YUKSEK COZUNURLUKLU EKRA
NDAKI METIN",1,2,8
70 TEXT 20,180,"[A]HERHANGI BIR YER [B]BEGEN
DIN!",1,1,16
80 PAUSE 5
90 NRM
```

SONUÇ: Program çalıştırıldığında, rastgele bir dizi çizgi çizilir. Ardından iki mesaj görüntülenir; birincisi normal boyutlarının sekiz katı büyüklüğünde karakterler kullanılarak küçük harflerle, ikincisi ise harfler arasında çift boşluk kullanılarak büyük ve küçük harflerle yazılır.

YEDİNCİ BÖLÜM
EKRAN MANİPÜLASYONU

7.1 GİRİŞ

SIMONS' BASIC, düşük çözünürlüklü grafikler ve ekran verisi işleme konusunda çeşitli kapsamlı özellikler içermektedir.

BCKGNDS komutu, bir karakterin arka plan rengini tanımlamanıza olanak tanır. FLASH komutu, tanımlanmış bir renkteki karakterleri belirli bir aralıkta ters alana ve ardından tekrar normal alana geçirir. BFLASH, ekranı çevreleyen kenarlığı benzer şekilde yanıp söndürür. OFF, FLASH komutunu sonlandırır.

FCHR komutu, ekranın tanımlanmış bir alanının seçilen bir karakterle doldurulmasını sağlar. FCOL komutu, ekranın belirtilen bir bölümünü belirlenmiş bir renkle doldurur. FILL komutu ise bu komutları birleştirerek, ekranın tanımlanmış bir alanını belirli bir karakterle ve belirli bir renkle doldurmanızı sağlar.

MOVE komutu, ekranın belirtilen bir bölümünü başka bir ekran konumuna kopyalar. INV komutu ise ekranın belirtilen bir bölümündeki karakterleri tersine çevirmenizi, yani normal karakterleri ters alan karakterlerine ve tam tersine dönüştürmenizi sağlar.

COPY komutu, Commodore seri yazıcınızı kullanarak grafik ekranının içeriğini yazdırmanıza olanak tanır. HRDCPY ise normal ekran verileri için aynı işlevi yerine getirir.

SCRSV komutu düşük çözünürlüklü bir ekran görüntüsünü kaydetmenizi sağlar. SCRLD ise kaydedilmiş bir ekran görüntüsünü geri çağırmanıza ve görüntülemenize olanak tanır.

Yedinci Bölüm ayrıca, ekranın tanımlanmış bir alanını belirlenmiş bir yönde kaydırmak için komutlar içerir. Bu, harflerin tanımlanmış alanın bir tarafından kayarak diğer taraftan tekrar görünmesiyle (sarmalama) veya harflerin tanımlanmış alandan kayarak tekrar görünmemesiyle (boşaltma) yapılabilir.

Bu bölümdeki komutların doğrudan modda veya bir programın parçası olarak kullanılabileceğini unutmayın.

7.2 BCKGNDS

FORMAT: BCKGNDS sc,b1,b2,b3

AMAÇ: Bir karakterin arka plan rengini değiştirmek.

Klavyedeki her karakter ekranda görüntülendiğinde, ekranda 8x8 piksellik bir kare kaplar. (Piksel, ekrandaki en küçük adreslenebilir noktadır.) Karenin rengi normalde ekranın geri kalanının rengiyle aynıdır (karakter ters alanda görüntülenmiyorsa elbette). BCKGNDS komutu, bu karenin rengini hem normal ekran hem de ters alan için değiştirmenize olanak tanır. BCKGNDS komutuyla yalnızca her tuşun üstünde yazılı olan karakterlerin kullanılabileceğini unutmayın. Grafik karakterler KULLANILAMAZ.

BCKGNDS komutunun sc parametresi ekranın rengini tanımlar. Sonraki üç parametre sırasıyla kaydırılmış bir karakterin, ters alanda kaydırılmamış bir karakterin ve ters alanda kaydırılmış bir karakterin arka plan rengini belirtir.

ÖRNEK: Üç farklı karakter arka plan rengi kullanarak bir mesaj yazdırmak için: (Aşağıdaki programda italik karakterlerin SHIFT tuşu basılı tutularak yazılması gerektiğini unutmayın.)

GİR:

```
10 PRINT"□"
20 BCKGNDS 1,3,5,6
30 PRINT"BU, SIMON BASIC":PRINT
40 PRINT"□BCKGNDS KOMUTUNA":PRINT
50 PRINT"□BIR ORNEKTIR": PRINT
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Üç satır metin görüntülenir; birincisi camgöbeği, ikincisi yeşil ve üçüncüsü mavi arka plan üzerindedir.

7.3 FLASH

FORMAT: FLASH renk,hız

Veya: FLASH renk

AMAÇ: Ekranda renk yanıp sönmesini sağlamak.

FLASH komutu, belirli bir ekran rengini normal ve ters alan gösterimi arasında dört saniyede bir veya tanımlanmış bir hızda değiştirmenizi sağlar. Bu tanımlanmış hız 1 ile 255 arasında olabilir. Her hız birimi yaklaşık olarak saniyenin altmışta biridir ve başlatıldıktan sonra, OFF komutu (bir sonraki bölüme bakın) kullanılana kadar yanıp sönme devam eder. FLASH komutunun yüksek çözünürlüklü veya çok renkli grafik ekranlarda kullanılamayacağını unutmayın.

ÖRNEK: Ekrandaki kırmızı renkteki alanların belirli bir hızda yanıp sönmesini sağlamak:

GİR:

```
10 PRINT"□"
20 PRINT AT(12,10)"OKUL,■ NEDEN■ KAPALI"
30 FLASH 2,10
1000 GOTO 1000
```

YAZ: **RUN <RETURN>**

SONUÇ: "OKUL" kelimesi yaklaşık her saniyenin altıda birinde sürekli olarak yanıp sönüyor.

ÖRNEK: Ekrandaki siyah renkli alanların her dört saniyede bir yanıp sönmesini sağlamak için:

EYLEM: Önceki bölümde yer alan programı girin ve ardından ekranda listeleyin.

KOMUT: **FLASH 0 <RETURN>**

SONUÇ: Program listesi her dört saniyede bir yanıp sönüyor.

7.4 OFF

FORMAT: OFF

AMAÇ: FLASH komutunu kapatmak için.

OFF komutu FLASH komutunu sonlandırır. Yanıp sönen karakterlerin renginin, OFF komutunun ne zaman kullanıldığına bağlı olduğunu unutmayın.

ÖRNEK: Yukarıdaki programda FLASH komutunu devre dışı bırakmak için:

GİR:

```
10 PRINT"■"
20 PRINT AT(12,10)"OKUL,■ NEDEN■ KAPALI"
30 FLASH 2,10
40 PAUSE 10
50 OFF
60 PRINT AT(12,10)"OKUL,■ NEDEN■ KAPALI"
```

YAZ: `RUN` **<RETURN>**

SONUÇ: "OKUL" kelimesi yalnızca on saniye boyunca yanıp sönüyor.

7.5 BFLASH

FORMAT: BFLASH hız,c1,c2

Veya: BFLASH 0

AMAÇ: Ekran çerçevesinin yanıp sönmesini sağlamak veya kapatmak.

BFLASH komutu, COMMODORE 64 ekranını çevreleyen çerçeveyi yanıp söndürmenizi sağlar. Komuttaki ilk parametre, 1 ile 255 arasında yanıp sönme hızını belirtir. Her birim yaklaşık olarak saniyenin altmışta biridir. c1 ve c2 parametreleri, çerçevenin yanıp söneceği renklerin numaralarıdır. BFLASH 0 yanıp sönmeyi kapatır. Çerçevenin ortaya çıkan renginin, komutun ne zaman yürütüldüğüne bağlı olduğunu unutmayın.

ÖRNEK: Kenarlığı önce kırmızı sonra mavi renkte yanıp sönmesini sağlamak için:

KOMUT: `BFLASH 25,2,6` **<RETURN>**

SONUÇ: Ekran çerçevesi sürekli olarak yanıp sönüyor, önce kırmızı, sonra mavi renkte ve yaklaşık her üçte bir saniyede bir değişiyor.

ÖRNEK: Ekran kenarını yanıp söndürmek ve ardından yanıp sönmeyi kapatmak için:

GİR:
```
10 BFLASH 25,2,6
20 PAUSE 5
30 BFLASH 0
1000 GOTO 1000
```

YAZ: **RUN <RETURN>**

SONUÇ: Çerçeve beş saniye boyunca önce kırmızı, sonra mavi renkte yanıp sönüyor.

7.6 FCHR

FORMAT: FCHR r,c,w,d,kod

AMAÇ: Ekranın bir alanını bir karakterle doldurmak.

FCHR komutu, ekranın tanımlanmış bir alanını belirli bir karakterle doldurmanızı sağlar. r ve c parametreleri, doldurulacak ekranın başlangıç noktasının satır ve sütun koordinatlarıdır. Satırlar 0'dan 24'e, sütunlar ise 0'dan 39'a kadar numaralandırılmıştır. w ve d parametreleri sırasıyla ekran alanının genişliğini ve derinliğini tanımlar. Genişlik karakter sayısı, derinlik ise satır sayısı cinsinden ölçülür. Son komut parametresi, görüntülemek istediğiniz karakterin 'poke' kodudur. (Poke kodlarının tam listesi COMMODORE 64 Kullanım Kılavuzunuzda yer almaktadır.)

ÖRNEK: 'A' harflerinden oluşan bir blok görüntülemek için:

GİR:
```
10 PRINT"□"
15 FCOL 0,0,10,10,0
20 FCHR 0,0,10,10,1
30 GOTO 20
```

YAZ: **RUN <RETURN>**

SONUÇ: Ekranın sol üst köşesindeki onar onluk bir blok A harfleriyle dolduruldu.

7.7 FCOL

FORMAT: FCOL r,c,w,l,renk

AMAÇ: Bir karakterin rengini değiştirmek.

FCOL komutu, belirtilen ekran alanındaki tüm karakterlerin rengini tanımlanmış bir renge değiştirir. FCHR komutunda olduğu gibi (önceki bölüme bakın), ilk dört komut parametresi kullanmak istediğiniz ekran alanını tanımlar. Son parametre, bu alanda görünen her karakter için yeni rengin numarasıdır. (Renk numaralarının listesi bu kılavuzun 6.3. bölümünde yer almaktadır.)

ÖRNEK: Karakterin rengini siyahtan kırmızıya değiştirmek için:

GİR:

```
10 PRINT"■"
15 FCOL 12,15,10,10,0
20 FCHR 12,15,10,10,1
30 FCOL 12,15,5,5,2
```

YAZ: RUN <RETURN>

SONUÇ: Yüz adet A harfinden oluşan bir blok sergileniyor. Bunların dörtte biri kırmızı, geri kalanı siyah renkte.

7.8 FILL

FORMAT: FILL r,c,w,l,kod,renk

AMAÇ: Ekranda tanımlanmış bir alanı belirli bir renkteki belirli bir karakterle doldurmak.

FILL komutu, ekranın tanımlanmış bir alanını belirli bir renk ve türdeki karakterlerle doldurmanıza olanak tanır. FCHR komutunda olduğu gibi (Bölüm 7.6'ya bakınız), FILL komutundaki ilk dört parametre, kullanılacak ekran alanını tanımlar. Sonraki parametre, görüntülenecek karakterin poke kodudur. (Poke kodlarının listesi COMMODORE 64 Kullanım Kılavuzunuzda yer almaktadır.) FILL komutundaki son parametre ise karakteri görüntülemek istediğiniz renktir.

ÖRNEK: Düz renk bloklarını görüntülemek için:

GİR:

```
10 REM"*** FILL ICIN ORNEK***
20 PRINT"■"
30 X=INT(40*RND(1)):Y=INT(25*RND(1))
40 V=INT(20*RND(1))+1:Z=INT(15*RND(1))+1
50 IF X+V>40 OR Y+Z>25 THEN 30
60 FILLY,X,V,Z,160,INT(16*RND(1)):GOTO30
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Ekranda rastgele konumlarda farklı renklerde bloklar görüntülenir.

EYLEM: **<RUN/STOP>** tuşunu basılı tutun ve **<RESTORE>** tuşuna basın.

SONUÇ: Normal ekran görüntülenir.

7.9 MOVE

FORMAT: MOVE r,c,w,l,dr,dc

AMAÇ: Ekranın bir bölümünü çoğaltmak.

MOVE komutu, ekranın tanımlanmış bir bölümünü ekranın başka bir yerinde yeniden görüntülemenizi sağlar. İlk dört komut parametresi, yeniden üretmek istediğiniz ekran alanını tanımlar (bkz. Bölüm 7.6). Son iki parametre ise, bilginin yeniden üretileceği alanın sol üst köşesinin satır ve sütun koordinatlarını belirtir.

NOT

Çoğaltılacak ekran alanının derinliği, bilgilerin çoğaltılacağı alanın satır numarasına eklendiğinde 25'i (ekran yüksekliğini) geçmemelidir. Benzer şekilde, çoğaltılacak alanın genişliği ile verilerin çoğaltılacağı alanın sütun numarası toplamı 40'ı (ekran genişliğini) geçmemelidir. Bu toplamları aşarsanız, "BAD MODE" mesajı görüntülenecek ve MOVE komutunu tekrar girmeniz gerekecektir.

ÖRNEK: Bir metin bloğunu çoğaltmak için:

GİR:

```
10 REM"*** MOVE ICIN ORNEK ***
20 PRINT"■BOSLUK TUSUNA BASIN"
30 PRINT"■EKRANIN"
40 PRINT"■BASKA BIR BOLUME"
50 PRINT"■BU ALANI TASIMAK ICIN."
60 GET A$:IF A$ <> " " THEN 60
70 MOVE 0,0,17,5,15,20
80 GOTO 80
```

YAZ: `RUN` **<RETURN>**

EYLEM: Boşluk tuşuna basın.

SONUÇ: Ekranın sol üst köşesindeki mesaj, sağ alt köşede de tekrar gösteriliyor.

7.10 INV

FORMAT: INV r,c,w,l

AMAÇ: Belirtilen ekran alanını tersine çevirmek.

INV komutu, tanımlanmış bir ekran alanındaki tüm normal karakterlerin ters alanda görüntülenmesine neden olur. Zaten ters alanda görüntülenen herhangi bir karakter normal şekilde görüntülenecektir. (Daha deneyimli programcılar için not: INV komutu, karakterin 7. bitini ayarlar veya temizler.)

ÖRNEK: Bir metin bloğunu tersine çevirmek için:

GİR:

```
10 REM"*** INV ICIN ORNEK ***"
20 PRINT"␣BOSLUK TUSUNA BASIN"
30 PRINT"BUNU TERSINE CEVIRMEK ICIN"
40 PRINT"EKRAN ALANI!"
50 GET A$:IF A$ <> " " THEN 50
60 INV 0,0,19,4
70 GOTO 50
```

YAZ: `RUN` **<RETURN>**

EYLEM: Boşluk tuşuna basın.

SONUÇ: Gösterilen mesaj ters yazı tipinde görüntülenir.

EYLEM: Boşluk tuşuna tekrar basın.

SONUÇ: Mesaj normal şekilde görüntüleniyor.

7.11 SCROLLING

FORMAT: direction W,sr,sc,ec,r

Veya: direction B,sr,sc,ec,er

AMAÇ: Ekranın bir alanını kaydırmak.

SIMON BASIC, ekran verilerinin belirli alanlarını dört yönden birinde kaydırmanıza olanak tanıyan bir komut sağlar. Kaydırma komutundaki ilk parametre, kaydırma yönünü belirtir - SOL, SAĞ, YUKARI veya AŞAĞI.

İkinci komut parametresi, sırasıyla 'ekranı sar' veya 'boş bırakma' ile kaydırmayı belirtmek için W veya B'dir. Ekranın bir bölümü 'ekranı sar' ile kaydırılırsa, belirtilen ekran alanındaki tüm karakterler bu alanın kenarından dışarı kayar ve karşı kenarda yeniden görünür. 'Boş bırakma', 'ekranı sar'dan farklı olarak, ekrandan dışarı kaydırılan karakterlerin yeniden görünmemesi anlamına gelir.

Kaydırma komutundaki `sr` ve `sc` parametreleri, kaydırmak istediğiniz alanın başlangıç satır ve sütun koordinatlarını tanımlar. `ec` ve `er` parametreleri ise kaydırma alanının bitiş sütun ve satır koordinatlarını belirtir. Farklı ekran alanlarını farklı yönlerde kaydırmak için kaydırma komutları birleştirilebilir. Herhangi bir kaydırma alanının maksimum yüksekliği ve genişliği sırasıyla 24 satır aşağı ve 23 karakter yataydan fazla olamaz. Yüksek çözünürlüklü veya çok renkli grafik ekranlarda kaydırma kullanılamayacağını unutmayın.

ÖRNEK: Ekranın iki bölgesini farklı yönlerde kaydırmak için:

GİR:

```
10 PRINT"␣"
20 FOR X = 0 TO 39
30 Y = INT(10 * SIN(X/π)) + 12
40 PRINT AT(X,Y)"*"
50 NEXT
60 LEFTW 0,0,20,25: RIGHTW 0,20,20,25
70 GOTO 60
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Ekranda aynı anda her iki yönde de kavisli bir çizgi kaydırılıyor.

7.12 EKRAN VERİLERİNİN SAKLANMASI VE GERİ ÇAĞIRILMASI

7.12.1 SCRSV

FORMAT: SCRSV 2,8,2,"isim,S,W"

Or: SCRSV 1,1,1,"isim"

AMAÇ: Düşük çözünürlüklü ekranlardan veri depolamak.

SCRSV komutu, düşük çözünürlüklü bir ekrandaki verileri kaset veya diskete kaydetmenizi sağlar. Komutu takip eden ilk rakam mantıksal dosya numarasıdır. Bu, COMMODORE 64'e disk sürücüsüne veya kaset ünitesine bir kanal açmasını söyler. İkinci rakam, kullanmak istediğiniz depolama aygıtını belirtir. Bu sayı kaset için 1 veya disket için 8'dir. Üçüncü rakam ikincil adrestir. Bu, bilgisayara bilgileri nasıl saklayacağını söyleyen özel bir talimattır. Örneğin, kaset için 1 ikincil adresi, COMMODORE 64'e bir dosya yazılacağını ve dosya kapatıldığında bandın sonuna bir dosya sonu işareti yerleştirileceğini bildirir. 'Ad', ekran verilerine vermek istediğiniz başlıktır. Bu ad, sakladığınız her ekran için benzersiz olmalıdır. Daha sonra bu adı, saklanan verileri geri çağırmak ve görüntülemek için SCRLD komutunda (bir sonraki bölüme bakın) kullanabilirsiniz. S parametresi, erişilen dosyanın sıralı olduğunu gösterir. W, COMMODORE 64'e bu dosyanın okunmak yerine yazılacağını bildirir. Kaydedildiğinde, her ekran yaklaşık beş blok yer kaplar. Parametrelerin virgülle ayrıldığına ve isim ile S,W'nin tırnak içinde belirtildiğine dikkat edin.

SCRSV komutu yüksek çözünürlüklü veya çok renkli grafikleri depolamak için kullanılamaz.

ÖRNEK: Fransız üç renkli bayrağını çizmek ve diskete kaydetmek:

GİR:

```
10 PRINT"□"
20 FILL 6,10,7,12,160,6
30 FILL 6,17,7,12,160,1
40 FILL 6,24,7,12,160,2
50 SCRSV 2,8,2,"UCRENK,S,W"
80 GOTO 80
```

YAZ: RUN <RETURN>

SONUÇ: Bayrak çizilir ve ardından diskete kaydedilir.

7.12.2 SCRLD

FORMAT: SCRLD 2,8,2,"isim"

Veya: SCRLD 1,1,0,"isim"

AMAÇ: Kaydedilmiş ekran verilerini geri çağırmak için.

SCRLD komutu, SCRSV komutuyla (önceki bölüme bakın) kaydedilmiş bir ekranı geri çağırmanıza ve görüntülemenize olanak tanır. Komuttan sonra gelen ilk rakam mantıksal dosya numarasıdır. Bu, COMMODORE 64'e disk sürücüsüne veya kaset ünitesine bir veri kanalı açmasını söyler. Komuttan sonra gelen ikinci rakam, verilerin depolandığı aygıtı belirtir. Bu sayı kaset için 1, disket için 8'dir. Üçüncü rakam ikincil adrestir. Bu, bilgisayara bilginin, depolanmadan önce kapladığı aynı bellek alanına yüklenmesi gerektiğini söyleyen özel bir talimattır. Ekran verilerine atadığınız başlık son parametredir ve tırnak işaretleri içinde olmalıdır.

ÖRNEK: Önceki bölümde açıklanan programda diskete kaydedilen ekran verilerini geri çağırmak ve görüntülemek için:

YAZ: `SCRLD 2,8,2,"UCRENK"` <RETURN>

SONUÇ: Üç renkli bayrak disketten geri çağrılır ve ekranda yeniden çizilir.

7.13 YAZDIRMA EKRANI VERİLERİ

7.13.1 GİRİŞ

SIMON BASIC, seri yazıcı kullanarak normal veya grafik ekranlardan bilgi kopyalamanıza olanak tanıyan iki komut sunar. Bu komutlar, grafik tasarımında veya istatistiksel gösterimde grafik ve histogram üretmede son derece kullanışlıdır.

7.13.2 COPY

FORMAT: COPY

AMAÇ: Grafik ekranının basılı bir kopyasını oluşturmak.

COPY komutu, bir grafik ekranının içeriğini seri yazıcıya yazdırır. Ekranda mükemmel daireler çizmek için CIRCLE komutunu (Bölüm 6.5.9'a bakın) kullandıysanız, yazıcıda aynı görüntüyü elde etmek için bu daireler için tanımladığınız yarıçapları değiştirmeniz gerektiğini unutmayın. Mükemmel bir daire yazdırmak için x yarıçapı y yarıçapına eşit olmalıdır. Ekranı tekrar görüntülemek için x yarıçapını orijinal değerine geri değiştirmeniz yeterlidir.

ÖRNEK: Yüksek çözünürlüklü bir ekranda bozuk bir pasta grafiği görüntülemek ve ardından yazıcıda yuvarlak bir grafik oluşturmak:

GİR:

```
10 HIRES 0,1:MULTI 5,4,6
20 CIRCLE 80,100,78,78,1
30 ANGL 80,100,120,78,78,1
40 ANGL 80,100,160,78,78,1
50 ANGL 80,100,220,78,78,1
60 ANGL 80,100,330,78,78,1
70 PAINT 90,35,1
80 PAINT 90,120,3
90 PAINT 80,110,2
105 LOW COL 5,4,7
110 PAINT 40,40,3
120 COPY
1000 GOTO 1000
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Ekranda düzleştirilmiş bir pasta grafiği görüntülenir ve ardından doğru bir daire yazdırılır.

7.13.3 HRDCPY

FORMAT: HRDCPY

AMAÇ: Düşük çözünürlüklü bir ekran görüntüsünün çıktısını almak.

HRDCPY komutu, düşük çözünürlüklü bir ekran görüntüsünü seri yazıcıda yazdırmanıza olanak tanır. Bu komut, form, fatura vb. yazdırmada oldukça kullanışlıdır.

ÖRNEK: Bir mesajı önce ekranda, sonra yazıcıda yazdırmak için:

GİR:

```
10 PRINT"□"
20 PRINT AT(5,8)"SIMON BASIC":PRINT
30 PRINT"BASIC AIDS'IN EN UST DUZEYI"
40 HRDCPY
50 END
```

YAZ: `RUN` **<RETURN>**

SONUÇ: Veriler ekranda görüntülenir ve ardından Commodore yazıcıda yazdırılır.

SEKİZİNCİ BÖLÜM
YARATIK VE KULLANICI TANIMLI GRAFİKLER

8.1 GİRİŞ

Sekizinci Bölüm, yaratıkların oluşturulması ve animasyonu ile kullanıcı tanımlı grafiklerin oluşturulmasıyla ilgili komutları içerir. Bölüm, bu konuların her biri için birer olmak üzere iki kısma ayrılmıştır.

8.2 YARATIKLAR

8.2.1 GİRİŞ

Yaratık, çeşitli şekillere dönüştürülebilen programlanabilir bir nesnedir. Bu nesne, bilgisayara yaratığın nereye yerleştirilmesi gerektiğini söyleyerek ekranda hareket ettirilebilir. (Yaratıklar hakkında daha ayrıntılı bir açıklama için COMMODORE 64 Kullanım Kılavuzuna bakabilirsiniz.)

SIMON BASIC'te bir yaratığa 'hareketli nesne bloğu' veya MOB denir. Ekranda aynı anda sekiz adede kadar bağımsız MOB görüntülenebilir ve hareket ettirilebilir. MOB'lar normal ve grafik ekranlarda görüntülenebilir. İki tür MOB vardır: yüksek çözünürlüklü ve çok renkli. Yüksek çözünürlüklü bir MOB, 24 nokta genişliğinde ve 21 nokta derinliğindedir. Bu tür MOB'daki her nokta bir piksel genişliğindedir. Çok renkli bir MOB, 12 nokta genişliğinde ve 21 nokta derinliğindedir. Burada her nokta iki piksel genişliğindedir. Yüksek çözünürlüklü bir MOB, 15 COMMODORE 64 renginden herhangi BİRİ ile boyanabilir. Çok renkli MOB'lar ise en fazla ÜÇ farklı renkle boyanabilir.

Standart BASIC'te, yaratıkların oluşturulması ve animasyonu birçok POKE komutu gerektirir. SIMON BASIC, POKE komutlarını basit, kullanımı kolay BASIC tipi komutlarla değiştirir.

DESIGN komutu, her bir MOB'un verilerinin depolandığı COMMODORE 64 belleğindeki konumu belirtmek için kullanılır. Her MOB daha sonra program listenizdeki bir ızgara üzerinde tasarlanır, böylece kullanılmadan önce şeklini görebilirsiniz. MOB'lar normal bir ekranda veya yüksek çözünürlüklü ve çok renkli grafiklerle birlikte kullanılabilir. MOB SET komutu, belirtilen bir MOB'u ayarlar ve birincil rengini atar. CMOB, çok renkli bir MOB tasarlarken kullanılmak üzere iki ek renk atamak için kullanılır. MMOB komutu, seçilen bir MOB'u belirtilen bir ekran konumuna görüntülemenizi ve/veya taşımanızı sağlar. RLOCMOB, görüntülenen bir MOB'u bir ekran konumundan diğerine taşımanızı sağlar. DETECT ve CHECK komutları, bir MOB'un başka bir MOB veya ekran verisi öğesiyle çarpışıp çarpışmadığını belirlemek için kullanılır.

Bu bölümdeki komutların yalnızca bir programın parçası olarak kullanılabileceğini unutmayın.

Bu kılavuzun bu bölümünde kullanılan örneklerin tümü, ekranda iki MOBS nesnesi görüntüleyen eksiksiz bir program oluşturmaya yöneliktir. Bu nedenle, örnekler arasında NEW komutunu kullanmayın ve programı çalıştırmanız söylenene kadar çalıştırmayın.

8.2.2 DESIGN

FORMAT: DESIGN c,ad

Veya: DESIGN c,sa + gc

AMAÇ: Bir MOB için bellek alanı ayırmak.

DESIGN komutu, oluşturduğunuz MOB için COMMODORE 64'ün belleğinde yeterli alan ayırır. Her MOB 64 bayt bellek kullanır. DESIGN komutundaki ilk parametre, MOB'un yüksek çözünürlüklü mü yoksa çok renkli mi olduğunu belirtir. Bu konumda "1" çok renkli, "0" ise yüksek çözünürlüklü modu gösterir. İkinci parametre olan ad, MOB verilerinin ilk baytının başlangıç adresini tanımlar. Bu sayı, 2048 ile 16319 arasında 64'ün katı olmalı ve ondalık veya onaltılık biçimde girilebilir. Onaltılık bir sayının önüne dolar işareti ($) konulmalıdır. Bir MOB yüksek çözünürlüklü bir grafik ekranında kullanılacaksa, bu rakama 49152 ondalık veya $C000 onaltılık grafik sabiti değeri eklemeniz gerekir.

Kullanılabilir MOB belleğinin her 64 baytlık alanı Blok olarak adlandırılır. MOB veri başlangıç adresini 64'e bölerseniz, Blok Numarası elde edersiniz. Bu numara, MOB'u kurmak için MOB SET komutunda (Bölüm 8.2.3'e bakın) kullanılır.

NOT

Grafik sabit rakamı, blok numarası hesaplanırken başlangıç adresine EKLENMEMELİDİR.

COMMODORE 64'ün hafızasında MOB verileri için mevcut alanlar ve ilgili blok numaraları aşağıda listelenmiştir:

BLOK NUMARALARI	BELLEK KONUMLARI
32-63	2048 - 4095
128 – 255	8192 - 16383

Eğer MEM komutunu kullandıysanız (Bölüm 8.3.1'e bakınız), MOB verileri için yalnızca 192 ile 255 arasındaki bloklar kullanılabilir durumdadır.

NOT

COMMODORE 64'ün belleğinin kaldırabileceği kadar çok MOB oluşturabilirsiniz. Ancak, aynı anda en fazla sekiz MOB görüntüleyebilirsiniz. Bir program sırasında bir MOB'u kaldırmak ve yerine başka bir MOB oluşturmak isterseniz, yeni MOB'u, değiştirdiğiniz MOB'un başlangıç adresini kullanarak tasarlamanız yeterlidir.

ÖRNEK: Normal bir ekranda yüksek çözünürlüklü bir MOB için bellek alanı ayırmak için:

GİR:
```
90 DESIGN 0,2048
```

SONUÇ: Çok renkli MOB oluşturulduğunda, verileri 32. Blok'ta 2048. bellek konumundan itibaren, yani 2048'in 64'e bölünmesiyle elde edilen değerden itibaren saklanır.

ÖRNEK: Normal bir ekranda çok renkli bir MOB için bellek alanı ayırmak için:

GİR:
```
320 DESIGN 1,2112
```

SONUÇ: Yüksek çözünürlüklü MOB oluşturulduğunda, verileri 33. Bloktaki 2112 numaralı bellek konumundan, yani 2112'nin 64'e bölünmesiyle elde edilen değerden saklanır.

8.2.3 @

FORMAT: @.....

Veya: @.....

AMAÇ: Bir MOB için tasarım ızgarasını oluşturmak.

@ komutu, bir MOB tasarımı için bir ızgara oluşturmanıza olanak tanır. Yüksek çözünürlüklü MOB'lar için ızgara 24 nokta genişliğinde, çok renkli MOB'lar için ise 12 nokta genişliğindedir. Her iki durumda da ızgara 21 satır derinliğindedir.

NOT

Izgaradaki her satır numarasının aynı uzunlukta, yani iki veya dört basamaklı olduğundan emin olun. Bunu yaparak, ızgaranın bir kısmının girintili olmasını önleyecek ve böylece MOB tasarım sürecini kolaylaştıracaksınız.

Bölüm 8.1'de açıklandığı gibi, yüksek çözünürlüklü MOB'lar için bir renk, çok renkli MOB'lar için ise üç renk kullanılabilir. Yüksek çözünürlüklü MOB rengi ve çok renkli bir MOB'un ana rengi, MOB SET komutunda tanımlanır (bir sonraki bölüme bakın). İki ek çok renkli MOB rengi, CMOB komutuyla atanır (Bölüm 8.2.4'e bakın). MOB üzerindeki her nokta için renkler, aşağıdaki tabloda yer alan karakterlerden biri kullanılarak atanır:

YÜKSEK ÇÖZÜNÜRLÜKLÜ MOB'LAR

RENK KODU	KULLANILAN RENK
B	MOB SET komutunda atanan renk

ÇOK RENKLİ MOBLAR

RENK KODU	KULLANILAN RENK
B	CMOB komutundaki Renk 1
C	MOB SET komutunda atanan renk
D	CMOB komutundaki Renk 2

Elbette, her iki MOB türünde de karakter girmeden ekran arka plan rengini de kullanabilirsiniz.

ÖRNEK: Yüksek çözünürlüklü bir MOB tasarlamak için:

GİR:

```
5 PRINT"□"
10 REM"*** MOB'LAR ICIN ORNEK ***"
80 REM"*** MOB'LARI TASARLAYIN ***"
90 DESIGN 0,2048
100 @.........BBBBB.........
110 @........BB...BB........
120 @........BB.....BB......
130 @........BB...BB........
140 @.........BBBBB.........
150 @............B..........
160 @....BBBBBBBBBBBBBBB....
170 @....BBBBBBB.BBBBBBB....
180 @....BBBBBB..BBBBBBB....
190 @...BBB..BBB.BBB..BBBB...
200 @...BBB..BBB.BBB..BBBB...
210 @...BBBBBBB...BBBBBBB...
220 @..BBBBBBBBBBBBBBBBBBB..
230 @..BBB...B...B..BBB.BBB..
240 @..BBBB.BB.B.B.B.BB.BBB..
250 @.BBBBB.BB.B.B.BB.B.BBBB.
260 @.BBBBB.BB...B.BBB..BBBB.
270 @.BBBBBBBBBBBBBBBBBBBBB.
280 @.BBBBBBBBBBBBBBBBBBBBB.
290 @.........................
300 @.........................
```

SONUÇ: Programın bu bölümü çalıştırıldığında, ızgara içindeki çizim, 32 numaralı bellek bloğunda MOB verisi olarak saklanır.

ÖRNEK: Çok renkli bir MOB tasarlamak için:

GİR:

```
320 DESIGN 1,2112
400 @......BB....
410 @.....BCDB...
420 @....BBCCCBBB
430 @...BBCCCB...
440 @.....BCB....
450 @.BB..BCB....
460 @BCCB.BCB....
470 @.BCB.BCB....
480 @..BB.BCB....
490 @...BBCCCB...
500 @..BDCDCDCB..
510 @.BDCDCDCDCB.
520 @.BCDCDCDCCB.
530 @..BCDCDCBB..
540 @...BBBBB....
550 @....B..B....
560 @...B....B...
570 @..B......B..
580 @...B......B.
590 @..BBB....BBB
600 @.....B.....B
```

SONUÇ: Programın bu bölümü çalıştırıldığında, ızgara içindeki çizim, 33 numaralı bellek bloğunda MOB verisi olarak saklanır.

8.2.4 CMOB

FORMAT: CMOB d,c2

AMAÇ: Çok renkli bir MOB için renkleri ayarlamak.

CMOB komutu, çok renkli bir MOB için iki ek renk tanımlamanıza olanak tanır; bunlar, MOB ızgarasında B ve D harfleriyle çizilen noktaların renkleridir (bkz. Bölüm 8.2.3).

ÖRNEK: Yukarıdaki programa devam ederek, çok renkli MOB'a siyah ve yeşil renkleri atayın:

GİR:

```
610 CMOB 0,5
```

SONUÇ: Çok renkli MOB görüntülendiğinde, B ile çizilen tüm noktalar siyah, D ile çizilen tüm noktalar ise yeşil olur.

8.2.5 MOB SET

FORMAT: MOB SET mb,blk,col,pr,res

AMAÇ: Bir MOB kurmak

Adından da anlaşılacağı gibi, MOB SET komutu bir MOB'u başlatır. mb parametresi, kurmakta olduğunuz MOB'un numarasını belirtir. Bu numara her MOB için benzersiz olmalıdır. MOB numarası ne kadar düşükse, diğer MOB'lara göre önceliği o kadar yüksek olur; yani, ekranda iki veya daha fazla MOB hareket ediyorsa, daha düşük numaralı bir MOB, daha yüksek numaralı bir MOB'un önüne geçer.

MOB SET komutunun ikinci parametresi olan blk, MOB verilerinin alınacağı bellek bloğunu tanımlar (bkz. Bölüm 8.2.2). Sonraki parametre olan col, ana MOB rengini, yani yüksek çözünürlüklü modda B ile veya çok renkli modda C ile çizilen MOB üzerindeki her noktaya atanacak rengi tanımlar.

`pr` parametresi, MOB'un ekran verilerine göre önceliğini belirtir; yani MOB'un ekrandaki diğer karakterlerin ÜZERİNDEN mi yoksa ALTINDAN mı geçmesini istediğinizi belirler. Bu konumda "0", MOB'a ekran verilerine göre öncelik verir; "1" ise ekran verilerine MOB'lara göre öncelik verir. MOB SET komutundaki son parametre olan `res`, MOB'un çok renkli veya yüksek çözünürlüklü modda oluşturulup oluşturulmadığını gösterir. Bu konumda "1", çok renkli modu; "0" ise yüksek çözünürlüklü modu belirtir.

ÖRNEK: Önceki bölümde anlatılan programda yüksek çözünürlüklü MOB dosyasını kurmak için:

GİR:
```
700 MOB SET 0,32,0,1,0
```

SONUÇ: Programın bu bölümü yürütüldüğünde, bellek bloğu 32'deki yüksek çözünürlüklü MOB numarası 0 başlatılır. Görüntülendiğinde, MOB siyah renkte olur ve tüm ekran verilerini üzerinden geçirir.

ÖRNEK: Önceki bölümde anlatılan programda çok renkli MOB'u kurmak için.

GİR:
```
710 MOB SET 1,33,2,0,1
```

SONUÇ: Programın bu kısmı yürütüldüğünde, bellek bloğu 33'te 1 numaralı çok renkli MOB oluşturulur. Görüntülendiğinde, MOB'un ana rengi kırmızıdır ve tüm ekran verilerini aktarır.

8.2.6 MMOB

FORMAT: MMOB mn,x1,y1,x2,y2,genişleme,hız

AMAÇ: Bir MOB'u görüntülemek ve/veya taşımak.

MMOB komutu, bir MOB'u ekranın bir noktasında görüntülemenize ve ardından isterseniz başka bir konuma taşımanıza olanak tanır. İlk parametre olan mn, görüntülemek ve taşımak istediğiniz MOB'un numarasını belirtir. x1 ve x2 parametreleri, MOB'un taşınmadan önce görüntüleneceği ekran noktasının koordinatlarıdır. x2 ve y2 parametreleri, hareket gerçekleştikten sonra MOB'un hedef noktasını gösterir. Bir MOB'u taşımak değil, yalnızca görüntülemek istiyorsanız, başlangıç ve bitiş ekran konumları için aynı koordinatları kullanmanız yeterlidir.

Genişleme, MOB'un görüntülendiğinde sahip olacağı boyutu ifade eder. Genişleme sayıları ve sonuç olarak elde edilen görüntüleme boyutları aşağıdaki tabloda gösterilmiştir:

GENİŞLEME	SONUCU
0	MOB normal boyutta görüntülenir.
1	MOB, x ekseninde genişletilmiştir, yani normal genişliğinin iki katı olarak görüntülenir.
2	MOB, y ekseninde genişletilmiştir, yani normal yüksekliğinin iki katı olarak görüntülenir.
3	MOB her iki eksende de genişletilir, yani normal genişliğinin ve yüksekliğinin iki katı olarak görüntülenir.

Hız, MOB'un hareket edeceği oranı belirtir. Bu sayı 1 ile 255 arasında olmalıdır: 1 en hızlı hız, 255 en yavaş hızdır.

ÖRNEK: Yukarıdaki programda MOB'ları taşımak için:

GİR:

```
800 MMOB 1,0,0,200,200,2,20
810 MMOB 0,0,0,185,70,3,20
```

SONUÇ: Programın bu bölümü çalıştırıldığında, yüksek çözünürlüklü MOB, y ekseninde genişletilmiş olarak ekranın üst kısmında görüntülenir. Çok renkli MOB ise normal boyutunun iki katı büyüklüğünde ekranın alt kısmında görünür.

8.2.7 RLOCMOB

FORMAT: RLOCMOB mn,x,y,genişleme,hız

AMAÇ: Bir MOB'u iki ekran konumu arasında taşımak.

RLOCMOB komutu, ekranda görüntülenen bir MOB'u farklı bir konuma taşımanızı sağlar. x ve y parametreleri, MOB'un taşınacağı noktanın ekran koordinatlarıdır. Diğer parametreler, MMOB komutunda kullanılanlarla aynıdır (önceki bölüme bakın).

ÖRNEK: Yukarıdaki programda yer alan her iki MOB'u da yeniden konumlandırmak için:

GİR:

```
820 FOR I=1 TO 20:X=150 * INT(RND(1))+50
830 RLOCMOB 1,X,200,2,10
840 RLOCMOB 0,X-15,70,3,10
850 NEXT
```

SONUÇ: Programın bu bölümü çalıştırıldığında, iki canavar ekranda birbirlerini kovalıyormuş gibi görünür.

8.2.8 DETECT

FORMAT: DETECT n

AMAÇ: MOB çarpışma algılama işlemini başlatmak için.

DETECT komutu, MOB çarpışma algılamasını açar. n'ye 0 değeri atanması, COMMODORE 64'ün bir MOB ile diğeri arasında çarpışma algılamasını sağlar. Komut parametresi olarak 1 kullanılırsa, MOB'lar ve ekran verileri arasındaki çarpışma algılaması başlatılır. DETECT komutunun her zaman İKİ KEZ kullanılması gerektiğini unutmayın. Komut ilk olarak, çarpışmanın gerçekleşip gerçekleşmediğini gösteren bilgisayar belleğindeki alanı temizlemek için kullanılır. (Bu alana 'yaratık çarpışma kaydı1' denir.) Komut ikinci kez kullanıldığında, çarpışma algılaması başlatılır.

ÖRNEK: Yukarıdaki programda yaratık çarpışma kaydını temizlemek için:

GİR:

```
825 DETECT 0
```

8.2.9 CHECK

FORMAT: IF CHECK (mn1,mn2) = 0 THEN eylem

Or: IF CHECK (0) = 0 THEN eylem

AMAÇ: MOB çakışmasını kontrol etmek için.

CHECK komutu, MOB'lar arasında veya bir MOB ile ekran verileri arasında çakışma olup olmadığını test etmek için kullanılır. Çakışma testi yapmak istediğiniz MOB'lar, komutun ardından gelen parantez içinde belirtilir.

Köşeli parantez içindeki sıfır değeri, COMMODORE 64'ün herhangi bir sprite ile ekran verisi arasında çarpışma olup olmadığını kontrol etmesine neden olur. Çarpışma meydana gelirse, tanımlanan işlem gerçekleştirilir.

ÖRNEK: Yüksek çözünürlüklü MOB'u ekranda aşağı kaydırarak, çok renkli MOB ile çakışma olup olmadığını kontrol etmek için:

GİR:

```
858 FOR P = 70 TO 200
859 DETECT 0:IF CHECK(0,1) = 0 THEN 865
860 RLOCMOB 0,X - 15,P,3,10: NEXT
```

SONUÇ: Programın bu bölümü çalıştırıldığında, bir MOB diğer MOB'a dokunduğunda 865. satırda belirtilen işlem (bir sonraki bölüme bakınız) gerçekleştirilir.

8.2.10 MOB OFF

FORMAT: MOB OFF mn

AMAÇ: Ekrandan bir MOB'u temizlemek.

MOB OFF komutu, ekrandaki bir MOB'u siler. mn parametresi, kaldırmak istediğiniz MOB'un numarasını belirtir.

ÖRNEK: Yukarıdaki programı tamamlamak için:

GİR:

```
855 PRINT AT(X/8 + 2,20)"VAY CA..."
856 PAUSE 1
865 PRINT AT(X/8 + 2,20)"VAY CANINA!!"
870 PAUSE 1
875 MOB OFF V,RLOCMOB 0,X-15,196,3,10
```

YAZ: RUN **<RETURN>**

EYLEM: Kuşu izleyin!

SONUÇ: Koşucu kuş, ağırlığın altında ezildi. (Tüm kuş severlerden özür dileriz.)

8.3 KULLANICI TANIMLI KARAKTERLER OLUŞTURMA

8.3.1 GİRİŞ

SIMON BASIC, mevcut klavye karakterlerini kendi tanımladığınız karakterlerle değiştirmenizi sağlayan bir özellik sunar.

COMMODORE 64 karakter seti ROM'da, yani Salt Okunur Bellekte (ROM) saklanır. Bu karakterleri yeniden tanımlamak için, karakter setinin RAM'e, yani Rastgele Erişim Belleğine (RAM) taşınması gerekir. MEM komutu bu işlevi gerçekleştirir. DESIGN komutu, yeniden tanımlamak istediğiniz karakteri poke kodu cinsinden belirtmenizi sağlar. (Poke kodlarının tam listesi COMMODORE 64 Kullanım Kılavuzunuzda yer almaktadır.) Her karakter bir ızgara içinde tasarlanır. Bu, karakterin oluşturulma aşamasını görüntülemenizi sağlar. Kullanıcı tanımlı karakterlerin grafik ekranında KULLANILAMAYACAĞINI unutmayın.

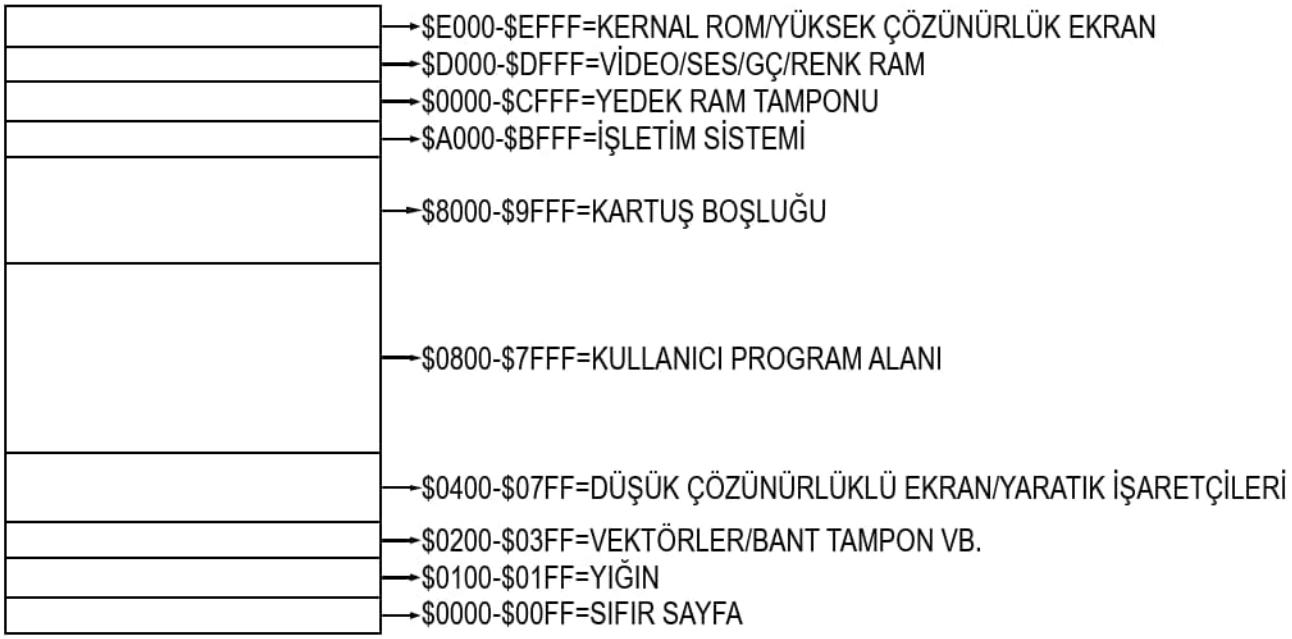
8.3.2 MEM

FORMAT: MEM

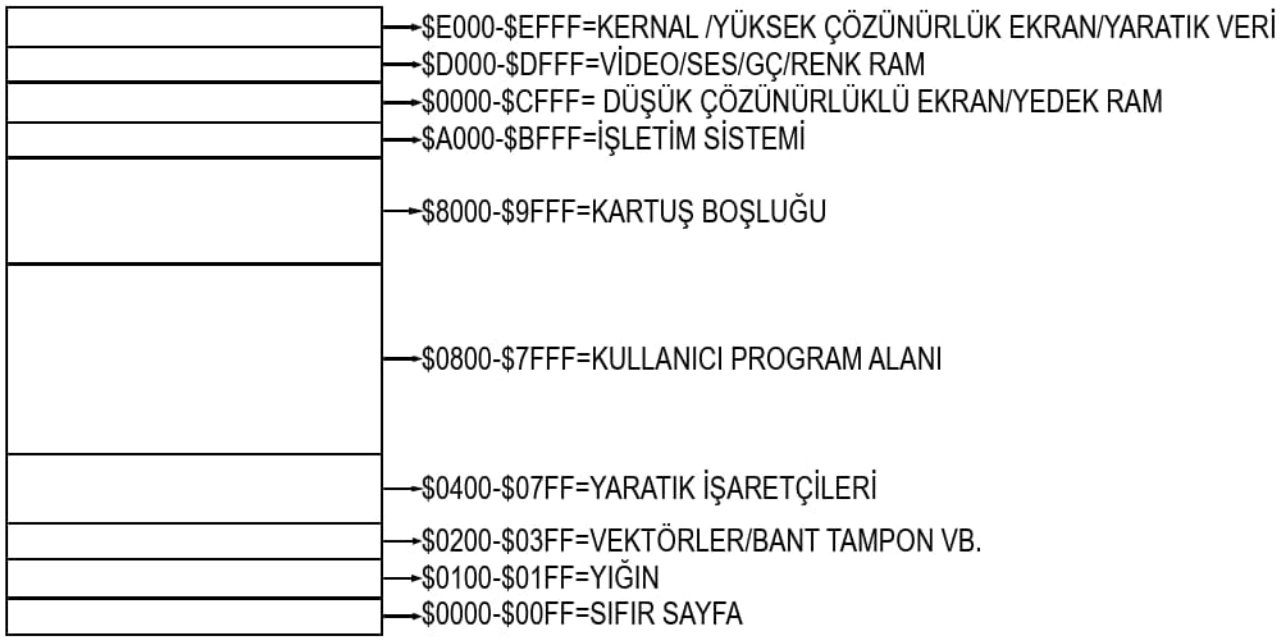
AMAÇ: Karakter ROM'unu RAM'e taşımak.

MEM komutu, ROM'daki karakter setini çekirdeğin arkasındaki RAM'e taşır. Ekran $CC00 konumuna taşınır ve yaratık verileri yalnızca $F000 konumundan, yani Blok 192'den itibaren eklenebilir (Bölüm 8.2.1'e bakın). Orijinal COMMODORE 64 karakter setine geri dönmek için, RUN/STOP tuşunu basılı tutun ve RESTORE tuşuna basın.

Şekil 8.1 ve 8.2, MEM komutu kullanılmadan önce ve sonra COMMODORE 64'ün bellek yapılandırmasını göstermektedir.



ŞEKİL 8-1 MEM ÖNCESİ BELLEK YAPILANDIRMASI



ŞEKİL 8-2 MEM SONRASI BELLEK YAPILANDIRMASI

NOT

Bir program MEM komutunu içeriyorsa, TRACE komutu (Bölüm 2.11.1'e bakınız) kullanılamaz.

ÖRNEK: Karakter ROM'unu RAM'e taşımak için:

GİR:
```
10 MEM
```

SONUÇ: Programın bu bölümü çalıştırıldığında, COMMODORE 64 karakter seti, karakterlerin yeniden tanımlanmasına hazırlık olarak RAM'e taşınır.

8.3.3 DESIGN

FORMAT: DESIGN 2,$E000 + ch * 8

AMAÇ: Kullanıcı tanımlı bir grafik karakterinin hangi karakterin yerini alacağını belirtmek için.

DESIGN komutu, kendi tanımladığınız bir karakterle değiştirilecek karakteri tanımlamanıza olanak tanır. Ayrıca, farklı bir format kullanan yaratık grafikleriyle ilişkili bir DESIGN komutu da bulunduğunu unutmayın (Bölüm 8.2.1'e bakınız).

Kullanıcı tanımlı karakterler yalnızca düşük çözünürlüklü ekranlarda kullanılabilir. DESIGN komutunun ardından gelen 2 rakamı, COMMODORE 64'e kullanıcı tanımlı karakter verilerinin geleceğini bildirir. Her yeni karakter 8 bayt bellek kaplar. Onaltılık sayı $E000, karakter verilerinin başlangıç adresidir. ch parametresi, değiştirmek istediğiniz mevcut karakterin poke kodudur. (Poke kodlarının listesini COMMODORE 64 Kullanım Kılavuzunuzda bulabilirsiniz.) Yeni karakter 8x8'lik bir ızgara içinde tasarlanır (bir sonraki bölüme bakın). Değiştirilen karakterin yazılı olduğu tuşu her kullandığınızda görüntülenir.

ÖRNEK: "Z" karakterini yeniden tanımlamak için:

GİR:
```
20 DESIGN 2,$E000 + 26 * 8
```

SONUÇ: Grafik karakter oluşturulduktan sonra, "Z" harfi her kullanıldığında görüntülenir.

8.3.4 @

FORMAT: @.....

@ komutu, kullanıcı tanımlı bir grafik karakterin tasarımı için bir ızgara oluşturmanıza olanak tanır. Izgara 8 nokta genişliğinde ve 8 satır derinliğindedir. Yeni karakter, ızgaradaki uygun noktanın üzerine 'B' harfi yerleştirilerek tasarlanır.

NOT

Izgaradaki her satır numarasının aynı uzunlukta, yani iki, üç veya dört basamaklı olduğundan emin olun. Bunu yaparak, ızgaranın bir bölümünün girintilemesini önleyecek ve karakter tasarım sürecini kolaylaştıracaksınız.

ÖRNEK: 'Silindir şapkalı' bir karakter tasarlamak için:

GİR:

```
20 DESIGN 2,$E000 + 26 * 8
30 @........
40 @........
50 @..BBBB..
60 @..BBBB..
70 @..BBBB..
80 @BBBBBBBB
90 @BBBBBBBB
95 @........
```

EYLEM: RUN **<RETURN>**

EKRAN:

```
READY.
```

EYLEM: **<Z>** tuşuna birkaç kez basın.

SONUÇ: 'Silindir şapkalı' karakter gösteriliyor.

Orijinal COMMODORE 64 karakter setine geri dönmek için, RUN/STOP tuşunu basılı tutun ve RESTORE tuşuna basın.

DOKUZUNCU BÖLÜM
YAPILANDIRILMIŞ PROGRAMLAMA

9.1 GİRİŞ

Standart BASIC'te programlama yaparken karşılaşılan ana sorunlardan biri, daha karmaşık programlarda yapılandırılmış bir akışın olmamasıdır. GOTO ve GOSUB komutlarının kullanımı, en basit program listeleri dışında kalan tüm kod listelerinin anlaşılmaz hale gelmesine neden olur - hatta programın yazarı için bile! Bu okunaksızlık, her bir rutinin ne yaptığını açıklamak için birden fazla REM ifadesi kullanılarak kısmen hafifletilebilir. Bu sadece zaman alıcı olmakla kalmaz, aynı zamanda çok fazla bellek alanı da tüketir.

SIMON BASIC, özel yapılandırılmış programlama komutlarıyla bu sorunları ortadan kaldırır. Bu komutlar, BASIC programlarınızda GOTO ve GOSUB komutlarına olan ihtiyacı büyük ölçüde ortadan kaldırır. Örneğin, PROC komutu, kullandığınız her program rutinini etiketlemek için kullanılır. (Bu işlev, COBOL'daki Prosedür Bölümündeki Paragraf adlandırmasına eşdeğerdir.) Bu rutinler, CALL veya EXEC komutları kullanılarak yürütülür.

FOR...NEXT döngülerinin yapısı da değiştirildi. REPEAT...UNTIL komutu, bir prosedürü tanımlanmış sayıda çalıştırmanıza olanak tanır. LOOP...EXIT IF...END LOOP, bir döngü içinde birden fazla koşul testi sağlar. Normal IF..THEN koşul testi artık, bir ifadenin önceden tanımlanmış bir koşulla eşleşip eşleşmemesi durumunda izlenecek yolların (aynı program satırında) belirtilmesini basitleştirmenizi sağlayan ELSE'i içerir. RCOMP komutu, kodu yeniden girmek zorunda kalmadan önceki IF..THEN...ELSE koşul testini kullanmanıza olanak tanır.

Bu bölümdeki komutların yalnızca bir programın parçası olarak kullanılabileceğini unutmayın.

9.2 DURUM TESTİ VE PROGRAM DÖNGÜLERİ

9.2.1 IF..THEN...ELSE

FORMAT: IF koşul THEN doğru:ELSE:yanlış.

AMAÇ: Bir koşulun olup olmadığını test etmek ve koşul doğruysa bir komuta, koşul yanlışsa başka bir komuta dallanmak.

IF..THEN...ELSE komutu, standart BASIC IF..THEN koşul testine benzer şekilde çalışır. Önemli bir fark, testin hem doğru hem de yanlış sonuçları için (aynı program satırında) kodun belirli bölümlerine dallanma yapılabilmesidir. ELSE ifadesinin her zaman önceki ve sonraki koddan iki nokta üst üste (:) ile ayrılması gerektiğini unutmayın.

ÖRNEK: Bir soruya verilen yanıtı kontrol etmek için:

GİR:

```
10 PRINT"COMMODORE BILGISAYARINIZ VAR MI
?"
20 PRINT "LUTFEN EVET VEYA HAYIR OLARAK
CEVAP VERIN (E/H)"
30 FETCH "□",1,A$
40 IF A$ = "E" THEN 60
50 IF A$ = "H" THEN 70:ELSE:GOTO 30
60 PRINT "TEBRIKLER":END
70 PRINT "GECMIS OLSUN":END
```

YAZ: RUN **<RETURN>**

EYLEM: İstendiğinde, E veya H tuşlarından birine ve ardından **<ENTER>** tuşuna basın.

SONUÇ: E veya H tuşuna basılmasına bağlı olarak uygun mesaj görüntülenir. Diğer tuşlara basılması herhangi bir işlem gerçekleştirmez.

9.2.2 REPEAT UNTIL

FORMAT: REPEAT döngü UNTIL koşul sağlanan kadar.

AMAÇ: Belirtilen koşul sağlanana kadar bir program döngüsünü çalıştırmak.

REPEAT....UNTIL, standart BASIC'teki FOR...NEXT döngüsüyle aynı işlevi görür; ancak döngünün başında kodun kaç kez çalıştırılacağını belirtmek yerine, döngü sayısı kodun sonundaki bir koşul testiyle belirlenir. REPEAT döngüyü başlatır; UNTIL ise bir koşulu test eder, örneğin X>10, bu koşul doğruysa program döngüden çıkar. Koşul sağlanmazsa, döngü yeniden çalıştırılır.

UYARI

DOKUZDAN FAZLA İÇ İÇE DÖNGÜ OLUŞTURAMAZSINIZ. BU SAYIYI AŞARSANIZ, "YIĞIN ÇOK BÜYÜK" MESAJI GÖRÜNTÜLENİR.

ÖRNEK: A'dan G'ye alfabenin harflerini yazdırmak için:

GİR:

```
10 A=65
20 REPEAT:PRINTCHR$(A):A=A+1:UNTIL A>70
30 PRINT"TAMAM!"
```

YAZ: `RUN` <RETURN>

EKRAN:
```
A
B
C
D
E
F
G
TAMAM!
```

9.2.3 RCOMP

FORMAT: RCOMP:doğru:ELSE:yanlış

AMAÇ: Son IF...THEN...ELSE koşul testini yeniden çalıştırmak.

RCOMP, bir programda en son tanımlanan IF...THEN...ELSE koşul testinin tekrarlanmasını sağlar. Bu, aynı kodu tekrar girme zorunluluğunu ortadan kaldırır.

ÖRNEK: Aynı koşul testini üç kez tekrarlamak için:

GİR:
```
10 INPUT A
20 IF A = 10 THEN PRINT "SELAM ";:ELSE:P
RINT "HOSCAKAL ";
30 RCOMP:PRINT "RECEP ";:ELSE:PRINT "YAB
ANCI ";
40 RCOMP:PRINT "HOSGELDIN":ELSE:PRINT "U
MARIM TEKRAR GORUSURUZ"
50 GOTO 10
```

YAZ: `RUN` <RETURN>

EKRAN: `?`

YAZ: `10` <RETURN>

EKRAN: `SELAM RECEP HOSGELDIN`

YAZ: `5` <RETURN>

EKRAN: `HOSCAKAL UMARIM TEKRAR GORUSURUZ`

SONUÇ: Bir rakam girildiğinde, bu giriş üç kez test edilir ve girilen değere uygun görüntü oluşturulur.

9.2.4 LOOP...EXIT IF...END LOOP

FORMAT: LOOP program döngü EXIT IF koşul doğru END LOOP

AMAÇ: Belirtilen koşul sağlanana kadar sürekli bir döngü gerçekleştirmek.

LOOP...EXIT IF... END LOOP komutu, REPEAT...UNTIL komutuna (Bkz. Bölüm 9.2.2) benzer şekilde bir program döngüsü gerçekleştirir, ancak önemli bir fark vardır. REPEAT...UNTIL yalnızca döngünün sonunda koşul testine izin verir. LOOP...EXIT IF ... END LOOP ise döngü içinde herhangi bir sayıda koşul testine izin verir. Bir koşul karşılanırsa, program END LOOP'tan sonraki ifadeye döner. Koşul karşılanmazsa, program döngüsü yeniden yürütülür.

UYARI

BEŞTEN FAZLA İÇ İÇE DÖNGÜ OLUŞTURAMAZSINIZ. BU SAYIYI AŞARSANIZ, "YIĞIN ÇOK BÜYÜK" MESAJI GÖRÜNTÜLENİR.

ÖRNEK: Klavyeden A ile F arasında bir karakter almak için:

GİR:

```
10 PRINT"A ILE F ARASINDA BIR HARF GIR"
20 LOOP
30 GET A$:IF A$ = "" THEN 30
40 EXIT IF ASC(A$)<65
50 EXIT IF ASC(A$)>70
60 PRINT A$;
65 END LOOP
70 PRINT CHR$(13)"ERISIM ALANI DISINDA":
END
```

YAZ: `RUN` **<RETURN>**

YAZ: `C` **<RETURN>**

EKRAN: `C`

YAZ: `B` **<RETURN>**

EKRAN: `CB`

YAZ: `S` **<RETURN>**

EKRAN: `ERISIM ALANI DISINDA`

SONUÇ: Klavye taranır ve tanımlanan aralıktaki tüm harfler ekranda görüntülenir. Aralığın dışında bir harf girilirse program döngüden çıkar.

9.3 PROGRAM PROSEDÜRLERİ

9.3.1 GİRİŞ

Daha yapılandırılmış kod yazımını kolaylaştırmak için SIMON BASIC, BASIC program rutinlerini etiketlemenizi ve gerektiğinde bu rutinleri adlarıyla çağırmanızı sağlayan dört komut sunar. Bu, büyük ölçüde programlarınızda GOTO ve GOSUB kullanma zorunluluğunu ortadan kaldırır. Bu rutinlere 'prosedürler' denir. Farklı programlarda sık sık kullanılan herhangi bir prosedür, bir prosedür 'kütüphanesinde' saklanabilir ve gerektiğinde yüklenebilir. PROC komutu (Bölüm 9.3.2'ye bakınız) prosedürlere ad atamak için kullanılır. Bunlar daha sonra CALL komutu (Bölüm 9.3.4'e bakınız) veya EXEC komutu (Bölüm 9.3.5'e bakınız) ile yürütülür. CALL komutu, standart BASIC GOTO komutuyla aynı şekilde çalışır, yani program prosedürün başına atlar ve yürütmeye o noktadan devam eder. EXEC komutu, GOSUB gibi davranır; yani program, belirtilen prosedüre atlar ve prosedür tamamlandığında EXEC komutunu takip eden program satırına geri döner. Prosedürün tamamlandığı her zaman END PROC komutuyla belirtilmelidir (bkz. Bölüm 9.3.3), bu komut standart BASIC'teki RETURN komutuyla aynı şekilde çalışır.

Bu kılavuzun bu bölümünde kullanılan örneklerin tümü, eksiksiz bir program oluşturmaya yöneliktir. Bu nedenle, örnekler arasında NEW komutunu kullanmayın ve program çalıştırılmadığı sürece çalıştırmayın.

UYARI

BEŞTEN FAZLA İÇ İÇE GEÇMİŞ İŞLEM OLUŞTURAMAZSINIZ. BU SAYIYI AŞARSANIZ, "YIĞIN ÇOK BÜYÜK" MESAJI GÖRÜNTÜLENİR.

9.3.2 PROC

FORMAT: PROC isim

AMAÇ: Bir program rutinini etiketlemek.

PROC komutu, program rutinlerini etiketlemenizi ve gerektiğinde bu rutinleri adlarıyla çağırmanızı sağlar. PROC komutunu takip eden satırdaki tüm karakterler, prosedürün adı olarak kabul edilir. Bu nedenle, PROC ve prosedür adının ardından aynı program satırında başka bir kod gelmemelidir.

ÖRNEK: Bir program rutinine "GIRIS ADI" etiketini atamak için:

GİR: 100 PROC GIRIS ADI

SONUÇ: Program çalıştırıldığında, bu satırı takip eden kod, GIRIS ADI adlı bir prosedür oluşturur.

9.3.3 END PROC

FORMAT: END PROC

AMAÇ: Bir işlemin sonunu belirtmek.

END PROC, 'kapalı' bir prosedürün, yani EXEC komutuyla çağrılacak bir prosedürün sonunu belirtir (Bölüm 9.3.5'e bakınız). Bu komut, standart BASIC'teki RETURN ile aynı şekilde çalışır; yani prosedür sona erdiğinde, program prosedürün çağrıldığı satırı takip eden satıra döner.

ÖRNEK: Kullanıcının adını bir değişkene kaydetmek için bir prosedür oluşturmak:

GİR:

```
100 PROC GIRIS ADI
110 PRINT "ISMINIZ NEDIR"
120 FETCH "S",15,A$
130 END PROC
```

SONUÇ: Program çalıştırıldığında, GIRIS ADI prosedürü EXEC komutu kullanılarak çağrılabilir (Bölüm 9.3.5'e bakınız).

9.3.4 CALL

FORMAT: CALL prosedür adı

AMAÇ: Program yürütmesini belirli bir kod satırına aktarmak.

CALL komutu, standart BASIC'teki GOTO komutuyla aynı şekilde çalışır, ancak satır numarası yerine bir prosedür adı kullanılır. Aynı program satırında CALL'dan sonra gelen her şey, çağrılan prosedürün adı olarak kullanılır. Bu nedenle, CALL ve prosedür adından sonra aynı program satırında başka bir kod gelmemelidir. Çağrılan prosedür 'açık' olmalıdır, yani END PROC içermeyen bir prosedür olmalıdır.

ÖRNEK: Sıralama programının başlangıcı:

GİR:

```
10 PRINT"C"
20 PRINT"KAC ISIM SIRALANACAK?"
30 PRINT"EN FAZLA 15 ISIM"
32 PRINT"VE HER ISIM ICIN"
34 PRINP"ON KARAKTERI GECMEYECEK"
40 FETCH"D",2,X
45 IF X > 10 AND X < 16 THEN DIM A$(X)
50 IF X > 0 AND X < 16 THEN CALL ADLAR
55 GOTO 10
60 PROC ADLAR
70 FOR I = 1 TO X
80 FETCH"S",10,A$(I):PRINTTAB(20)"TAMAM"
90 NEXT
```

SONUÇ: Girilen isim sayısı 16'dan az ise, işlem 60. satırdaki ADLAR prosedüründen devam eder.

9.3.5 EXEC

FORMAT: EXEC prosedür adı

AMAÇ: Bir program rutinini çağırmak ve işlem tamamlandığında çağrıdan sonraki satıra geri dönmek.

EXEC, standart BASIC'teki GOSUB ile aynı işlevi görür; yani program belirli bir kod bölümüne atlar, kodu yürütür ve END PROC'a (bkz. Bölüm 9.3.3) ulaşıldığında EXEC'ten sonraki satıra geri döner. Aynı program satırında EXEC'ten sonra gelen her şey, çağrılan prosedürün adı olarak kabul edilir. Bu nedenle, EXEC ve prosedür adından sonra aynı program satırında başka bir kod gelmemelidir.

ÖRNEK: Önceki bölümdeki Sıralama programını tamamlamak için.

GİR:

```
100 EXEC SIRALA
110 PRINT"[U]"
120 FOR I=1 TO X:PRINTTAB(20)A$(I):NEXT
130 END
140:
150:
1000 PROC SIRALA
1020 M=1
1030 REPEAT
1040 T = 0:FOR I = 1 TO M
1050 IF A$(I) < A$(I+1) THEN 1070
1060 W$=A$(I):A$(I)=A$(I+1):A$(I+1)=W$:T
=1
1070 NEXT I
1071 IF T = 1 THEN 1040
1080 M=M+1
1090 UNTIL M = X
1110 END PROC
```

YAZ: **RUN <RETURN>**

EYLEM: İstenildiğinde, her isim arasında RETURN tuşuna basarak en fazla 15 isim girin.

SONUÇ: İsimler sıralanır ve ardından görüntülenir.

9.4 PROGRAM DEĞİŞKENLERİ

9.4.1 GİRİŞ

Standart BASIC'te değişken kullanımı, farklı amaçlar için birçok değişkene ihtiyaç duyulduğunda kafa karıştırıcı hale gelebilir. SIMON BASIC, aynı değişkeni iki şekilde kullanmanıza olanak tanıyarak bu sorunu çözüyor: belirli bir program rutini içinde yerel olarak veya tüm program boyunca küresel olarak. Bu, ihtiyaç duyduğunuz değişken sayısını azaltır ve sonuç olarak daha fazla bellek alanı açarak daha uzun programlar yazmanıza ve çalıştırmanıza olanak tanır.

BASIC programındaki her değişkenin değeri, değişkenin ne zaman ve nerede kullanıldığına bağlı olarak değişir. LOCAL komutu, değişkenlerin şu anda sahip olduğu değerleri saklamanıza, bunları temizlemenize ve ardından aynı değişken adlarını kodun belirli bir bölümünde kullanmanıza olanak tanır. GLOBAL komutu ise LOCAL komutu çalıştırılmadan önce değişkenlerin sahip olduğu değerleri geri yükler.

9.4.2 LOCAL

FORMAT: LOCAL değişken1, değişken2, değişken3, ...

AMAÇ: Değişkenleri belirli bir program rutinine atamak.

LOCAL komutu, önceden tanımlanmış değişkenlerin değerlerini geçici olarak temizlemenize ve ardından bu değişkenleri belirli bir program rutini içinde yerel olarak kullanmanıza olanak tanır. GLOBAL komutu (bir sonraki bölüme bakın) değişkenlere orijinal değerlerini geri yükler.

UYARI

LOCAL KOMUTUYLA TANIMLANAN DEĞİŞKENLER ÖNCEDEN TANIMLANMIŞ OLMALIDIR. BU UYARIYA UYULMAMASI PROGRAMIN 'TAKILMASINA', YANİ DAHA FAZLA ÇALIŞMAMASINA NEDEN OLACAKTIR. PROGRAMDAN ÇIKMAK İÇİN RUN/STOP VE RESTORE TUŞLARINA BASMANIZ GEREKİR.

ÖRNEK: Değişkenlere yerel olarak değer atamak için:

GİR:

```
10 REM"*** LOCAL ICIN ORNEK ***
20 PRINT"␣"
30 A$="BASLANGIC DEGERI":A%=123:A=456.7
40 LOCAL A$,A%,A
50 A$="YENI DEGER ":A%=789:A=321.4
60 PRINT A$,A%,A
```

YAZ: `RUN` <RETURN>

EKRAN:
```
YENI DEGER 789 321.4
```

SONUÇ: Değişkenlere başlangıçta atanan değerler saklanır. Daha sonra değişkenlere yeni değerler atanır ve bu değerler yazdırılır.

9.4.3 GLOBAL

FORMAT: GLOBAL

AMAÇ: Yerel değişkenlere orijinal değerlerini geri yüklemek.

GLOBAL komutu, bir program rutini içinde yerel olarak kullanılan tüm değişkenlerin temizlenmesine neden olur. LOCAL komutu kullanılmadan önce (önceki bölüme bakın) sahip oldukları değerler daha sonra yeniden atanır.

ÖRNEK: Önceki bölümdeki programı kullanarak, küresel değerleri yerel olarak kullanılan değişkenlere geri yükleme:

GİR:
```
10 REM"*** LOCAL/GLOBAL ICIN ORNEK***
20 PRINT"␣"
30 A$="BASLANGIC DEGERI":A%=123:A=456.7
40 LOCAL A$,A%,A
50 A$="YENI DEGER ":A%=789:A=321.4
60 PRINT A$,A%,A
70 GLOBAL
80 PRINT A$,A%,A
```

YAZ: `RUN` <RETURN>

EKRAN:
```
YENI DEGER 789 321.4
BASLANGIC DEGERI 123 456.7
```

SONUÇ: LOCAL komutu kullanılmadan önce değişkenlere atanan değerler geri yüklendi.

ONUNCU BÖLÜM
HATA YAKALAMA

10.1 GİRİŞ

SIMON BASIC, BASIC programlarınızın 'çökmesini' önlemek için program hatalarını yakalamak üzere komutlar sağlar. ON ERROR komutu, bir hata bulunduğunda programda belirtilen bir noktaya dallanmanıza olanak tanır. ERRLN değişkeni, hatanın meydana geldiği program satırının numarasını, ERRN değişkeni ise hata numarasını içerir. ERRN'de tutulan değeri test ederek, isterseniz kendi hata mesajınızı oluşturup görüntülemek de dahil olmak üzere uygun işlemi gerçekleştirebilirsiniz. OUT, en son kullanılan ON ERROR komutunu devre dışı bırakır. NO ERROR komutu sizi normal COMMODORE 64 hata işleme rutinlerine geri döndürür.

10.2 ON ERROR

FORMAT: ON ERROR: GOTO satır numarası

AMAÇ: Program hatalarını yakalamak.

ON ERROR komutu, BASIC program hatalarını yakalamanıza olanak tanır. Bu sayede programlarınızın "çökmesini" önleyebilirsiniz. Bir hata bulunduğunda, program GOTO ile belirtilen satıra atlar. Hata numarası ERRN değişkeninde tutulur. Hatanın oluştuğu satır numarası ERRLN değişkeninde tutulur. ERRN'de tutulan değeri test ederek hangi hatanın oluştuğunu kontrol edebilir ve isterseniz kendi hata mesajınızı görüntülemek de dahil olmak üzere gerekli işlemleri yapabilirsiniz.

NOT

Belirli bir hatayı test ettikten ve uygun işlemi yaptıktan sonra, program yürütmesine devam etmeden önce her zaman OUT komutunu (aşağıdaki bölüme bakın) kullanmalısınız. Bu komut, ON ERROR komutunu içeren bir programı durdurduysanız ve kodunuzun bir kısmını düzenlemek istiyorsanız da kullanılmalıdır.

SIMON BASIC tarafından yakalanabilen hatalar ve ilgili hata numaraları aşağıda gösterilmiştir:

HATA NUMARASI	HATA
1	Too many files
2	File open
3	File not open
4	File not found_
5	Device not present
10	Next without for
11	Syntax
12	Return without gosub
13	Out of data
14	Illegal quantity
15	Overflow
16	Out of memory
17	Undefined statement
18	Bad subscript
19	Re-dimensioned array
20	Division by zero
21	Illegal direct
22	Type mismatch
23	String too long

Aşağıdaki örneklerde, lütfen bilgileri tam olarak gösterildiği gibi yazın. Yazım hataları kasıtlıdır ve SIMON BASIC hata yakalama komutlarının kullanımını göstermek amacıyla eklenmiştir.

ÖRNEK: Bir SYNTAX hatasını yakalamak ve kullanıcı tanımlı bir hata mesajı görüntülemek için:

GİR:

```
5 REM"*** HATA ISLEME ORNEGI ***"
10 ON ERROR: GOTO 100
15 PRIN"␣"
20 READ B
25 PRINT B: GOTO 20
30 DATA 1,2,3,4,5
100 IF ERRN =11 THEN PRINT"YAZIM HATASI
SATIR";ERRLN
```

YAZ: `RUN` **<RETURN>**

EKRAN:

```
YAZIM HATASI SATIR 15
```

SONUÇ: BASIC anahtar kelimesi PRINT'in yazımı yanlış olduğu için program 100. satırdaki hata işleme rutinine atladı. Programda bir SYNTAX hatası, yani 11 numaralı hata aranır. Bu hata oluştuğu için kullanıcı tanımlı hata mesajı görüntülenir.

EYLEM: **<RUN/STOP>** tuşuna basın.

EKRAN:
```
READY.
```

ÖRNEK: OUT OF DATA hatasını yakalamak için:

YAZ:
```
OUT <RETURN>
```

(OUT komutu aşağıdaki bölümde açıklanmıştır.)

EYLEM: Önceki bölümdeki program örneğinin 15. satırındaki yazım hatasını düzeltin ve ardından aşağıdaki kod satırını girin:

```
110 IF ERRN = 13 THEN PRINT"YETERLI VERI
 YOK SATIR";ERRLN
```

YAZ:
```
RUN <RETURN>
```

EKRAN:
```
1
2
3
4
5
YETERLI VERI YOK SATIR 20
```

SONUÇ: Sadece beş veri öğesi olduğundan, program altıncı öğeyi okuma girişiminde bulunulduğunda 100. satırdaki hata yakalama rutinine atlar. 110. satır, 13 numaralı hatayı, yani OUT OF DATA hatasını kontrol eder ve bu hata oluştuğu için tanımlanmış hata mesajını görüntüler.

10.3 OUT

FORMAT: OUT

AMAÇ: Son ON ERROR komutunu devre dışı bırakmak.

OUT komutu, en son kullanılan ON ERROR komutunu devre dışı bırakır. ON ERROR komutu tarafından yakalanan COMMODORE 64 hata işleme rutinine geri dönmek istiyorsanız, bu komut her zaman kullanılmalıdır.

ÖRNEK: Önceki bölümdeki program örneğini kullanarak, ON ERROR komutunu devre dışı bırakmak için:

EYLEM: Aşağıdaki revize edilmiş kod satırını girin:

```
19 J = J + 1: IF J = 6 THEN NO ERROR
20 READ B
25 PRINT B: GOTO 19
```

YAZ: `RUN <RETURN>`

EKRAN:
```
1
2
3
4
5
?OUT OF DATA ERROR
```

SONUÇ: Yalnızca dört veri öğesi olduğundan, 5 numaralı öğeyi okumaya çalışıldığında program çöker. Bu tür hataya ilişkin COMMODORE 64 mesajı görüntülenir.

10.4 NO ERROR

FORMAT: NO ERROR

AMAÇ: COMMODORE 64 hata işleme rutinlerini yeniden etkinleştirmek.

NO ERROR komutu, SIMON BASIC'in tüm hata yakalama komutlarını devre dışı bırakır ve kontrolü COMMODORE 64 hata işleme rutinlerine geri verir.

ÖRNEK: Önceki bölümdeki programı kullanarak, hata işleme kontrolünü COMMODORE 64'e geri döndürmek:

EYLEM: Aşağıdaki revize edilmiş kod satırlarını girin:

```
15 PRIN"␣"
100 NOERROR:IF ERRN = 11 THEN PRINT"YAZI
M HATASI SATIR";ERRLN
```

EYLEM: `RUN <RETURN>`

EKRAN:
```
SYNTAX ERROR IN LINE 100
```

ON BİRİNCİ BÖLÜM
SIMON BASIC İLE MÜZİK YAPMAK

11.1 GİRİŞ

COMMODORE 64'ün çekici özelliklerinden biri de müzik sentezleme yeteneğidir. Pratik ve deneyimle, birçok farklı müzik aletinin sesini yeniden üretmek mümkündür. COMMODORE 64'ün müzik ve ses özellikleri kapsamlıdır. SIMON BASIC, tüm bu özellikleri kullanmak üzere tasarlanmamıştır. Kartuş tarafından sağlanan ses ve müzik komutları, öncelikle sizi COMMODORE 64'te ses programlama sanatına tanıtmak amacıyla tasarlanmıştır. Commodore bilgisayarlarında müzik programlama becerilerinizi daha da geliştirmek istiyorsanız, lütfen yerel bayinizden bu konuyla ilgili özel yazılımlar ve kitaplar hakkında bilgi isteyin.

11.1.1 SES ŞEKİLLENDİRME

Farklı frekanslar farklı sesler üretir. Frekans ne kadar yüksekse, üretilen nota da o kadar yüksek olur. Çoğu kişisel bilgisayarda ses özelliği bulunur, ancak diğerlerinden farklı olarak COMMODORE 64 size her frekansı 'şekillendirme' olanağı sunar. Şekillendirme, bilgisayara bir frekansın her bölümünün nasıl çalınması gerektiğini söylemek anlamına gelir. Bir müzik notasının veya sesin yüksekliği, onu ilk duyduğunuz andan itibaren, artık duyamadığınız ve tamamen kaybolduğu ana kadar değişir. Tüm frekanslar dört döngüde üretilir: YÜKSELME, DÜŞME, DURMA ve KAYBOLMA (ATTACK, DECAY, SUSTAIN, RELEASE). Bu döngüler birlikte bir ses 'zarfı' oluşturur. Zarf içindeki her döngünün işlevi aşağıda açıklanmıştır:

YÜKSELME (ATTACK)

Bu, bir frekansın sıfırdan en yüksek ses seviyesine yükselme hızını belirler.

DÜŞME (DECAY)

Bu, frekansın en yüksek ses seviyesinden orta seviye bir ses seviyesine düşme hızını tanımlar.

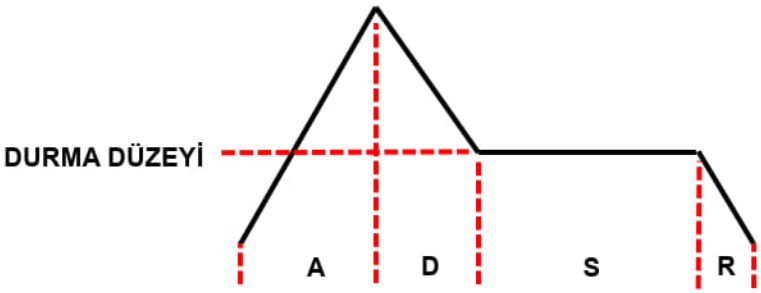
DURMA (SUSTAIN)

Bu, orta frekans ses seviyesini belirler.

KAYBOLMA (RELEASE)

Bu, frekansın SÜRDÜRÜLEBİLİR seviyesinden sıfır ses seviyesine düşme hızını belirler.

Açıklık sağlamak amacıyla, Şekil 11-1'de ses zarfını temsil eden bir diyagram gösterilmiştir.



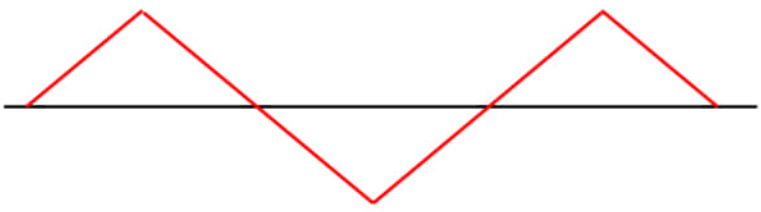
ŞEKİL 11-1 SES ZARFI

11.1.2 SES DALGALARI

COMMODORE 64, oluşturduğunuz müzik veya ses efektlerini çalmak için kullanmak istediğiniz ses dalgası türünü seçmenize olanak tanır. Her ses dalgası türü farklı bir etki üretir. Dalga biçimleri ve ürettikleri etkiler aşağıda açıklanmıştır. Her dalga biçiminin diyagramı da gösterilmiştir.

ÜÇGEN

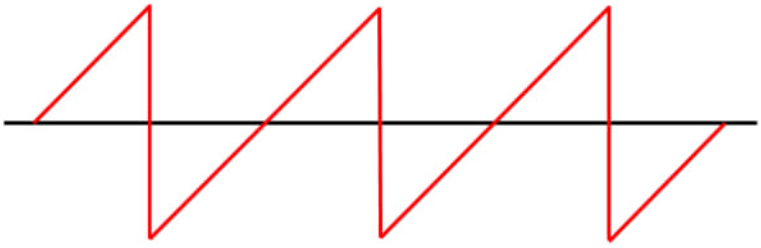
Bu dalga biçimi düşük harmonik içeriğine sahip ve yumuşak, flüt benzeri bir tınısı var.



ŞEKİL 11-2 ÜÇGEN SES DALGASI

TESTERE DİŞİ

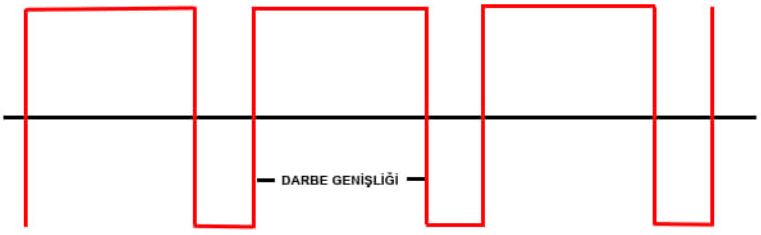
Testere dişi dalga formu, harmonikler açısından zengindir ve parlak, pirinç benzeri bir tınıya sahiptir.



ŞEKİL 11-3 TESTERE DİŞİ ŞEKLİNDE BİR SES DALGASI

DARBE/KARE

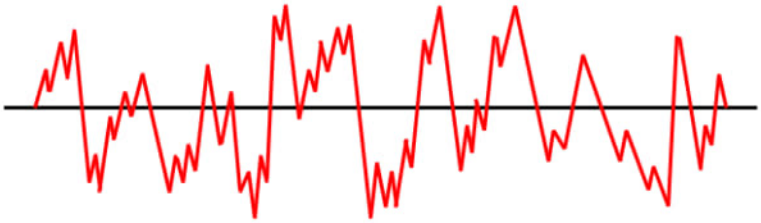
Darbe/Kare dalga formu, parlak, yankılı seslerden burunlu, tiz vuruşlara kadar çeşitli tonlar üretmenizi sağlar.



ŞEKİL 11-4 DARBE/KARE DALGA

GÜRÜLTÜ (NOISE)

Adından da anlaşılacağı gibi, bu dalga biçimi özel ses efektleri için çeşitli gürültü türleri üretir.



ŞEKİL 11-5 BİR GÜRÜLTÜ DALGASI

11.1.3 PROGRAMLAMA SESİ

Standart BASIC'te, COMMODORE 64'te müzik çalmak veya özel ses efektleri oluşturmak, birden fazla POKE komutu kullanmayı gerektirir. Bu, zahmetli ve zaman alıcı olabilir, ancak SIMON BASIC ile bu artık bir kısıtlama değildir. Kartuş tarafından sağlanan özel müzik ve ses komutları, POKE'larla bellek konumlarına kendiniz erişme ihtiyacını ortadan kaldırır.

VOL komutu, müziğinizin veya ses efektlerinizin ne kadar yüksek veya alçak sesle çalınacağını tanımlamanıza olanak tanır. WAVE komutu, sesleriniz için kullanmak istediğiniz dalga biçimini seçmenizi sağlar. ENVELOPE komutu, bir ses zarfı içindeki her notanın 'şeklini' tanımlamanıza olanak tanır. MUSIC komutu, üretmek istediğiniz sesleri oluşturmak için kullanılırken, PLAY komutu seslerin üretilmesini sağlar. Bu komutların yalnızca bir programın parçası olarak kullanılabileceğini unutmayın.

Aşağıdaki bölümlerde her bir SIMON BASIC müzik komutunun biçimi ve amacı açıklanmaktadır. Her komutun kullanımına dair kısa bir örnek de verilmiştir.

Bu kılavuzun bu bölümünde kullanılan örneklerin tümü, bir melodi çalan eksiksiz bir program oluşturmaya yöneliktir. Bu nedenle, örnekler arasında NEW komutunu kullanmayın ve programa çalıştırmanız söylenene kadar çalıştırmayın.

11.2 MÜZİK KOMUTLARI

11.2.1 VOL

FORMAT: VOL n

AMAÇ: Müzik ses seviyesini seçmek için.

VOL komutu, komutu takip eden müzik veya sesin çalınacağı ses seviyesini tanımlamanızı sağlar. Ses seviyeleri 0 ile 15 arasında değişir. 15 en yüksek ses seviyesidir ve 0 sesi kapatır. Yeni bir VOL komutu verilene kadar ses seviyesi ayarlanmış olarak kalır.

ÖRNEK: Ses seviyesini 15 olarak ayarlamak için:

GİR:
```
10 VOL 15
```

SONUÇ: Bu koddan sonra gelen herhangi bir ses en yüksek ses seviyesinde çalınacaktır.

11.2.2 WAVE

FORMAT: WAVE ses numarası, ikili sayı

AMAÇ: Müzik ses türünü ayarlamak.

WAVE komutu, müziğinizi veya ses efektlerinizi çalmak için kullanmak istediğiniz dalga biçimi türünü seçmenize olanak tanır. (Bkz. Bölüm 11.1.2.) WAVE komutundaki ilk parametre, sesin çalınacağı 'sesi' belirtir. COMMODORE 64'ün 1'den 3'e kadar numaralandırılmış üç sesi vardır. Her ses aynı dokuz oktavı içerir. Bu, bir sesi bir ses üzerinden çalabileceğiniz ve ardından başka bir sesten bir ses karıştırabileceğiniz anlamına gelir.

WAVE komutundaki ikinci parametre ikili bir sayıdır. (WAVE komutunda bu sayının başına dolar işareti konulmaz.) Bu sayı, COMMODORE 64'e her bir sesi nasıl çalacağını söyler. Sayının içindeki sekiz bitin her biri belirli bir işlevi yerine getirir. Bir işlevi seçmek için, ilgili bit ayarlanır, yani o konuma 1 yerleştirilir. Bitler 0'dan 7'ye kadar numaralandırılmıştır, 7. bit sayının en soldaki bitidir. Her bitin gerçekleştirdiği işlev aşağıdaki tabloda gösterilmiştir:

BİT NUMARASI	FONKSİYONU GERÇEKLEŞTİRİLDİ
0	Kapı bitini ayarlar (gerekli değil)
1	Senkronizasyonu ayarlar
2	Halka modülasyonunu ayarlar
3	Test bitini ayarlar (asla ayarlanmamalıdır)
4	Üçgen dalga formunu ayarlar
5	Testere dişi dalga formunu ayarlar
6	Darbe/Kare dalga formunu ayarlar
7	Gürültü

Bu işlevler aşağıda daha ayrıntılı olarak açıklanmıştır:

BİT 0 - KAPI BİTİ

SIMON BASIC yüklü olmayan bir COMMODORE 64'te, bu bit ayarlandığında Zarf Üretecini 'tetikler', yani bir frekansın dört döngüsünün başlamasına neden olur. Ancak, SIMON BASIC, PLAY komutu yürütüldüğünde bu biti otomatik olarak ayarladığından (Bölüm 11.2.5'e bakın), bu bitin değerini her zaman 0'da bırakmalısınız.

BİT 1 – SENKRONİZASYON

Senkronizasyon biti, bir sesten gelen bir notanın (frekansın) başka bir sesten gelen bir notayla senkronize edilmesini sağlar. Bir sesten sabit bir nota (statik frekans) çalarken, başka bir sesten birden fazla nota (değişken frekans) çalarak, çok çeşitli karmaşık armoniler üretilebilir. En iyi etki için, statik frekans her zaman değişken frekansın en düşük değerinden daha düşük olmalıdır. Değişken frekansı üretmek için seçilen ses, statik frekans için seçebileceğiniz sesi belirler. Bu, aşağıdaki tabloda özetlenmiştir:

DEĞİŞKEN FREKANSLI SES	STATİK FREKANSLI SES
1	3
2	1
3	2

Ses numarası, WAVE komutunda (yukarıya bakınız) ilk parametre olarak belirtilir.

BİT 2 - HALKA MODÜLASYONU

Bit 2 ayarlandığında, Halka Modülasyonunu başlatır. Bu efekt, Senkronizasyona (önceki bölüme bakın) benzer, ancak hem frekans hem de genlik (ses seviyesi) aynı anda değiştirilerek 'vay vay' efekti oluşturulabilir. Bir sesin frekansını başka bir sesin statik frekansına karşı değiştirerek, zil veya gonk sesleri ve özel efektler oluşturmak için çok çeşitli harmonik olmayan yapılar üretilebilir.

Halka Modülasyonunun duyulabilir olması için, değişken frekanslı ses için üçgen bir dalga formu seçilmelidir. Bu dalga formu daha sonra, bu ve tanımlanmış başka bir sesin çıkışının modüle edilmiş bir kombinasyonuyla değiştirilir. Senkronizasyonda olduğu gibi, değişken frekansı çıkarmak için seçilen ses, statik frekans için seçebileceğiniz sesi belirler. Bu, aşağıdaki tabloda özetlenmiştir:

DEĞİŞKEN FREKANSLI SES	STATİK FREKANSLI SES
1	3
2	1
3	2

BİT 3 - TEST BİTİ

Bu bit SIMON BASIC'te kullanılmaz. Bu nedenle asla ayarlanmamalı, yani her zaman 0 olmalıdır.

BİT 4 - ÜÇGEN DALGA FORMU

4. bitteki 1 değeri, üçgen dalga formu oluşturur.

BİT 5 - TESTERE DİŞİ DALGA ŞEKLİ

Bu bitteki 1 değeri, testere dişi dalga formu oluşturur.

BİT 6 - DARBE/KARE DALGA FORMU

6. bitteki 1 değeri, Darbe/Kare dalga formu oluşturur.

BIT 7 – GÜRÜLTÜ

Bu bitteki 1 değeri bir Gürültü dalga formu oluşturur.

NOT

4 ila 7. bitlerde, bir bit 1 olarak ayarlanırsa, kalan bitler 0 olarak bırakılmalıdır.

ÖRNEK: 1. ses için üçgen dalga formu oluşturmak.

GİR:
```
20 WAVE 1,00010000
```

SONUÇ: Programın bu bölümü çalıştırıldığında, komutu takip eden müzik üçgen dalga formu kullanılarak çalınır.

11.2.3 ENVELOPE

FORMAT: ENVELOPE vn,a,d,s,r

AMAÇ: Bir sesin 'şeklini' tanımlamak.

Bölüm 11.1.1'de açıklandığı gibi, COMMODORE 64, çalmak istediğiniz sesin şeklini belirleyen bir zarf tanımlamanıza olanak tanır. ENVELOPE komutu bu şekli tasarlamanızı sağlar. vn parametresi, sesi çalmak istediğiniz sesin numarasıdır. a, d ve r parametreleri sırasıyla üretilecek frekansın ATTACK, DECAY ve RELEASE döngülerinin sürelerini belirtir. ATTACK, DECAY ve RELEASE döngülerinin süreleri saniyenin binde biri biriminde ölçülür. Bu, 0 ile 15 arasında bir sayı ile temsil edilir. Bu sayılar ve karşılık gelen zaman döngüleri aşağıdaki tabloda listelenmiştir:

DEĞER (ZAMAN/DÖNGÜ)	YÜKSELME ORANI	DÜŞME KAYBOLMA (ZAMAN DÖNGÜSÜ)
0	2	6
1	8	24
2	16	48
3	24	72
4	38	114
5	56	168
6	68	204
7	80	240
8	100	300
9	250	750
10	500	1500
11	800	2400
12	1000	3000
13	3000	9000
14	5000	15000
15	8000	24000

Her müzik telinin sonunda, <SHIFT> tuşunu basılı tutun, <CLR/HOME> tuşuna basın ve G harfini girin. Bu, son notanın Bırakma döngüsünün (Bölüm 11.M'ye bakın) tetiklenmesine neden olur.

SUSTAIN parametresi (s), 0 ile 15 aralığındadır. Bu, sesin tutulacağı ara ses seviyesini tanımlar ve seçilen nota çalınırken VOL (bkz. Bölüm 11.2.1) tarafından ayarlanan ses seviyesini değiştirmeye eşdeğerdir. Ancak, bunun yalnızca seçilen notayı etkilediğini unutmayın. Diğer sesler aracılığıyla çalınan seslerin ses seviyesi değişmeden kalır.

ÖRNEK: 1. ses kanalından çalınan müzik için bir ses zarfı oluşturmak:

GİR:
```
30 ENVELOPE 1,8,8,8,0
```

SONUÇ: Bu komutu takip eden tüm müzik notaları, Saldırı, Sönme ve Bırakma döngüleri için eşit hızlarda ve 8'e ayarlanmış ara ses seviyesinde 1. ses üzerinden çalınır.

11.2.4 MUSIC

FORMAT: MUSIC n,"müzik dize"

Veya: MUSIC n,değişken + değişken + değişken....

AMAÇ: Müzik bestelemek veya ses efektleri oluşturmak.

MUSIC komutu, müzik bestelemenize ve çalmanıza veya ses efektleri oluşturmanıza olanak tanır. İlk komut parametresi, bir müzik vuruşunun süresini ifade eder. Bu sayı 1 ile 255 arasında olmalıdır - 1 en uzun süre, 255 en kısa süredir. Bu parametreyi takiben, çalmak istediğiniz müzik notaları dizisini tanımlarsınız. Notalar dizisi en fazla 255 karaktere kadar eklenebilir. Notaları çalmak istediğiniz ses, dizinin başında belirtilir. Bunu yapmak için, <SHIFT> tuşunu basılı tutun, <CLR/HOME> tuşuna basın (ters alanlı bir 'kalp' görüntülenir) ve ilgili ses numarasını girin. Bir notalar dizisinde yalnızca BİR ses kullanılabileceğini unutmayın.

Müzik notaları La'dan Sol'a kadar olan aralıktadır. Do, her oktavın ilk notasıdır, yani notaların sırası Do, Re, Mi, Fa, Sol, La, Si'dir. Bir müzik diyezi, SHIFT tuşuna basılı tutularak ve ilgili notanın harfine basılarak tanımlanır. Bir notayı bemol olarak çalmak istiyorsanız, önceki notayı diyezlemeniz gerekir, örneğin Mi bemol Re diyez ve Si bemol La diyez olarak girilir. Müzik susları Z harfiyle gösterilir.

Çalınacak notanın oktavı, 0 ile 8 arasında bir sayı ile tanımlanır. Bu sayı, notadan SONRA girilir. (Sus işaretleri elbette oktav numarası olmadan girilir.) Her notanın süresi, oktav numarasını takip eden bir kontrol karakteri ile belirtilir. Bu karakter, dört işlev tuşundan birine basılarak girilir. İşlev tuşları ve ilgili nota süreleri aşağıdaki tabloda gösterilmiştir:

FONKSİYON TUŞU	NOTA SÜRESİ
F1 TUŞU	Bir vuruşun on altıda biri
F3 TUŞU	Bir vuruşun sekizde biri
F5 TUŞU	Bir vuruşun dörtte biri
F7 TUŞU	Yarım vuruş
F2 TUŞU	Bir vuruş
F4 TUŞU	İki vuruş
F6 TUŞU	Dört vuruş
F8 TUŞU	Sekiz vuruş

Dizideki son notanın süresini belirledikten sonra, <SHIFT> tuşunu basılı tutun, <CLR/HOME> tuşuna basın ve G harfini girin. Bu, her notanın Bırakma döngüsünün (Bölüm 11.1.1'e bakınız) tetiklenmesine neden olur.

ÖRNEK: Bir melodi bestelemek:

GİR:

```
40 A$="■1Z<F1>C5<F1>E5<F1>F5<F1>"
50 A2$="G5<F7>C5<F1>E5<F1>F5<F1>G5<F7>C5
<F1>E5<F1>F5<F1>G5<F3>E5<F3>C5<F3>E5<F3>
D5<F5>E5<F1>E5<F1>D5<F1>C5<F7>C5<F1>"
60 A2$=A2$+"E5<F3>G5<F3>G5<F1>F5<F5>F5<F
3>E5<F1>F5<F1>G5<F3>E5<F3>C5<F3>D5<F3>C5
<F3>C5<F1>C5<F1>E5<F1>F5<F1>"
70 A3$="C5<F7>C5<F1>■G
80 MUSIC 8,A$ + A2$ + A2$ + A3$
```

SONUÇ: Programın bu bölümü yürütüldüğünde, notalar A$, A2$ ve A3$ değişkenlerinde saklanır ve bir melodi oluşturmak üzere 'birleştirilir'.

11.2.5 PLAY

FORMAT: PLAY n

AMAÇ: Bestelenmiş müzik çalmak.

Adından da anlaşılacağı gibi, PLAY komutu bestelediğiniz müziği çalmanıza olanak tanır. Komutu takip eden parametre, müziğin programın geri kalanıyla ilişkili olarak nasıl çalınacağını gösterir. Bu konumda "0" müzik çalmayı durdurur. "1" müzik çalar ve programın devam etmesinden önce bitmesini bekler. "2" müzik çalar ve programın yürütülmesine devam eder.

ÖRNEK: Bestelediğiniz müziği çalmak ve program yürütmesine devam etmek için:

GİR:
```
90 PLAY 2
100 GOTO 100
```

EYLEM: **RUN <RETURN>**

SONUÇ: 'Azizler Yürüyüşe Çıktığında' şarkısı çalınıyor.

ÖRNEK: Ses efekti oluşturmak için.

GİR:
```
10 VOL 15
20 WAVE 1,10000000
30 ENVELOPE 1,0,10,0,0
40 MUSIC 5,"▯1C5<F2>"
45 REPEAT
50 PLAY 1
55 A = A + 1: UNTIL A = 5
```

EYLEM: **RUN <RETURN>**

SONUÇ: Beş el ateş edildi.

PLAY 2'nin yüksek çözünürlüklü veya çok renkli grafiklerle birlikte kullanılamayacağını lütfen unutmayın.

ON İKİNCİ BÖLÜM

OKUMA FONKSİYONLARI

12.1 GİRİŞ

On İkinci Bölüm, SIMON BASIC kartuşu tarafından sağlanan dört okuma işlevini açıklamaktadır. Programlarınıza bir ışıklı kalem, joystick veya kumanda kolu eklediyseniz, bu işlevler bu cihazların konumunu belirlemenizi sağlayacaktır.

NOT

Işık kalemi SADECE oyun portu 1'e, yani AÇMA/KAPATMA düğmesinden en uzaktaki porta takılmalıdır. Joystick her iki oyun portuna da takılabilir. (Commodore 64 Kullanım Kılavuzuna bakın.)

PENX ve PENY komutları, bir ışık kaleminin ekranda işaret ettiği konumu belirlemenizi sağlar. POT komutu, bir kumanda kolunun ekrandaki konumunu okurken, JOY komutu bir joystickin hangi yöne işaret ettiğini belirlemenizi sağlar.

12.2 PENX

FORMAT: değişken = PENX

AMAÇ: Işık kaleminin x koordinatını döndürmek.

PENX fonksiyonu, ışıklı kalemin ekran üzerindeki konumunu, yani sol kenardan itibaren konumunu döndürür. Döndürülen sayı 0 ile 320 arasındadır. Bu fonksiyonun örneği aşağıdaki bölümde yer almaktadır. PENX değerinin her zaman PENY değerinden önce okunması gerektiğini unutmayın.

12.3 PENY

FORMAT: değişken = PENY

AMAÇ: Işık kaleminin y koordinatını döndürmek.

PENX fonksiyonu, ışıklı kalemin ekrandaki konumunu (yani yukarıdan aşağıya doğru) döndürür. Döndürülen sayı 0 ile 200 arasındadır.

ÖRNEK: Ekran üzerinde ışıklı kalemle çizim yapmak ve ardından çizimi yazdırmak:

EYLEM: Aşağıdaki programı girin ve ardından ÇALIŞTIRIN. Programın talimatları ekranda görüntülenecektir.

GİR:

```
10 REM:***ISIKLI KLEM PROGRAMI***
20 HIRES0,1
30 TEXT10,10,"'RUN' YAZDIKTAN SONRA",1,1
,8
40 TEXT10,25,"'KONTROL PORT 1'DEKI ISIKL
I KALEMI",1,1,8
50 TEXT10,40,"YERLESTIRMEK ICIN 15 SANIY
ENIZ VAR",1,1,8
60 TEXT10,55,"CIZIMINIZ TAMAMLANDIGINDA"
,1,1,8
70 TEXT10,70,"ISIKLI KALEMI KULLANICI PO
RTUNDAN CIKARIN",1,1,8
80 TEXT10,85,"VE YAZDIRMAK ICIN BOSLUK T
USUNA BASIN",1,1,8
90 PAUSE 15
100 HIRES0,1
110 LINE10,10,300,10,1:LINE300,10,300,18
0,1
120 LINE300,180,10,180,1:LINE10,180,10,1
0,1
130 GET A$:IF A$="" THEN 130
140 IF A$=" " THEN 160
150 PLOT (PENX+51)AND255,(PENY-50)AND255
,1:GOTO 150
160 COPY:END
```

YAZ: RUN **<RETURN>**

12.4 POT

FORMAT: değişken = POT(0)

Veya: değişken = POT(1)

AMAÇ: Kürek direncini geri kazandırmak.

POT fonksiyonu, bir küreğin ne kadar döndürüldüğünü belirlemenizi sağlar. Döndürülen sayı 0 ile 255 arasındadır. Komutu takip eden sayı, değeri okunacak olan küreği tanımlar.

ÖRNEK: Küreği kullanarak bir karakteri hareket ettirmek için:

GİR:

```
10 REM:****KUREK PROGRAM****
20 REM:******RECEP'TEN******
30 HIRES0,1
40 TEXT10,10,"KUREKLERI HAREKET ETTIRMEK
ICIN SUNLARI YAPABILIRSIN",1,1,8
50 TEXT10,25,"YARATIKLARI BULUSTURUN",1,
1,8
60 TEXT10,40,"NEDEN DENEMIYORSUN?",1,1,8
65 TEXT10,55,"(KUREK X=0 VE KUREK Y=255
OLARAK AYARLA)",1,1,8
70 DESIGN 0,64*32+49152
80 @.........................B
90 @.........................B
100 @.........................B
110 @...........BBBBBBBBBBBB
120 @...........B...........B
130 @...........B...........B
140 @...........B...........B
150 @...........B.......BBBBB
160 @...........B.......B...B
170 @...........B.......B...B
180 @.BBBBBBBBBBBBBBBBBBBBBB
190 @...........B.......B...B
200 @...........B.......B...B
210 @...........B.......BBBBB
220 @...........B...........B
230 @...........B...........B
240 @...........B...........B
250 @...........BBBBBBBBBBBB
260 @.........................B
270 @.........................B
280 @.........................B
290 DESIGN 0,64*33+49152
300 @B.......................
310 @B.......................
```

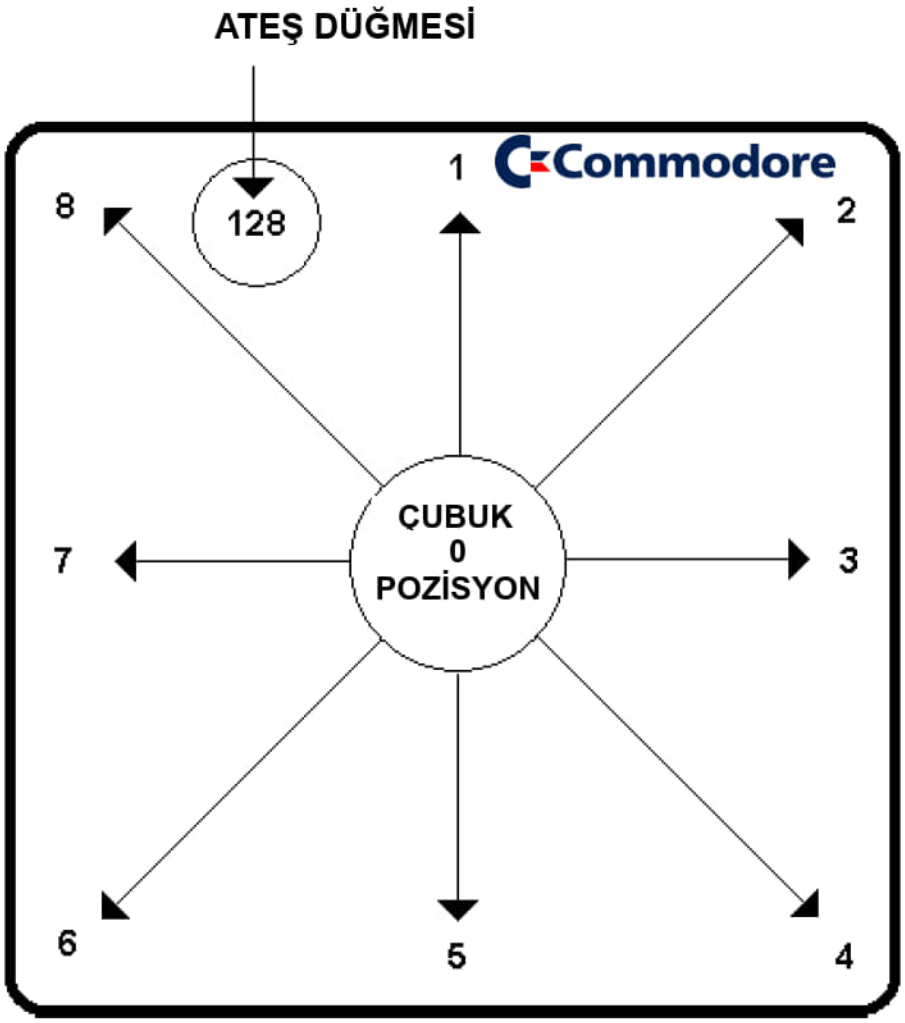
```
320 @B........................
330 @BBBBBBBBBBBBB...........
340 @B...........B...........
350 @B...........B...........
360 @B...........B...........
370 @BBBBB.......B...........
380 @B...B.......B...........
390 @B...B.......B...........
400 @BBBBBBBBBBBBBBBBBBBBBBB.
410 @B...B.......B...........
420 @B...B.......B...........
430 @BBBBB.......B...........
440 @B...........B...........
450 @B...........B...........
460 @B...........B...........
470 @BBBBBBBBBBBBB...........
480 @B........................
490 @B........................
500 @B........................
510 MOB SET 0,32,2,0,0
520 MOB SET 1,33,2,0,0
530 Z=10:W=245
535 MMOB0,Z,170,Z,170,3,50
536 MMOB1,W,170,W,170,3,50
540 X=INT(POT(0))
550 FORL=ZTOX
555 IFL=100THEN615
560 RLOCMOB0,L,170,3,50
580 NEXTL
590 Z=X:GOTO540
615 Y=INT(POT(1))
620 FORB=WTOY
625 IFB=145THEN700
630 RLOCMOB1,B,170,3,50
640 NEXTB
650 W=Y:GOTO615
700 TEXT105,170,"ATES",1,3,8:PAUSE1
703 MOB OFF 0:MOB OFF 1
705 TEXT105,170,"ATES",0,3,8
710 TEXT175,145,"BOOM !!!",1,5,20
716 BFLASH10,0,7
720 VOL15
730 WAVE 1,10008000
740 ENVELOPE 1,3,0,15,0
750 A$="␣1Z<F1>G5<F5>"
760 MUSIC 8,A$
770 PLAY1
820 VOL0:BFLASH0:HIRES0,1:PRINT"␣"
```

12.5 JOY

FORMAT: değişken = JOY

AMAÇ: Joystickin konumuna karşılık gelen değeri döndürmek.

JOY fonksiyonu, joystick'in hangi yöne işaret ettiğini veya ateşleme düğmesinin basılı tutulup tutulmadığını test etmenizi sağlar. Döndürülen değerler ve ilgili joystick konumları Şekil 12-1'de gösterilmiştir.



ŞEKİL 12-1 JOYSTICK DEĞERLERİ

ÖRNEK: Joystick ile bir şekil çizmek ve ardından boyamak:

EYLEM: Aşağıdaki programı girin ve ardından ÇALIŞTIRIN. Programın talimatları ekranda görüntülenecektir.

```
10 REM:*********JOYSTICK PROGRAMI******
11 HIRES0,1
12 LINE10,20,310,20,1:LINE310,20,310,160
,1
13 LINE310,160,10,160,1:LINE10,160,10,20
,1
15 TEXT88,30,"CIZ-DOLDUR",1,2,10
16 TEXT90,43,"──────────────────────",1,2,8
17 TEXT18,60,"SUNLARA DIKKAT EDEREK BIR
SEKIL CIZIN",1,2,10
18 TEXT26,80,"BOSLUK VERMEDEN SINIRLAR I
CINDE",1,2,10
19 TEXT60,100,"DEVAM EDIN.",1,2,10
20 TEXT105,140,"RECEP TARAFINDAN HAZIRLA
NDI",1,2,8:PAUSE3
30 HIRES0,1
33 TEXT93,5,"CIZ-DOLDUR",1,1,8
40 TEXT4,18,"NELER OLACAGINI GORMEK ICIN
",1,1,8
45 TEXT73,31,"BIR SEKIL CIZIN VE ATES DU
GMESINE BASIN ",1,1,8
50 LINE10,50,310,50,1:LINE310,50,310,150
,1
55 LINE310,150,10,150,1:LINE10,150,10,50
,1
80 X=160:Y=100
90 PLOTX,Y,1
100 IF JOY=1 THEN Y=Y-1:GOTO200
110 IF JOY=2 THEN Y=Y-1:X=X+1 : GOTO200
120 IF JOY=3 THEN X=X+1:GOTO200
130 IF JOY=4 THEN X=X+1:Y=Y+1 : GOTO200
140 IF JOY=5 THEN Y=Y+1:GOTO200
150 IF JOY=6 THEN Y=Y+1:X=X-1 : GOTO200
160 IF JOY=7 THEN X=X-1:GOTO200
170 IF JOY=8 THEN X=X-1:Y=Y-1 : GOTO200
180 IF JOY<>128 THEN 100
185 TEXT27,178,"IYI YAPTIN.",1,3,16
190 PAUSE2: GOTO 340
200 IF X<20 THEN X=20
300 IF X>300 THEN X=300
310 IF Y<60 THEN Y=60
320 IF Y>140 THEN Y=140
330 GOTO 90
340 LOW COL 2,1,1
350 PAINTX+1,Y+1,1:PAUSE5
360 HIRES0,1
370 TEXT50,50,"TEKRAR ICIN BOSLUK TUSU",
1,2,8
380 GETA$:IFA$=""THEN380
390 IFA$=" "THEN 30
400 GOTO380
```

ON ÜÇÜNCÜ BÖLÜM

SIMON BASIC PROGRAMLARINDAN ÖRNEKLER

13.1 GİRİŞ

On üçüncü bölüm, SIMON BASIC kartuşu kullanılarak neler yapılabileceğini göstermek için dört program içermektedir. Her programı yazıp ardından ÇALIŞTIRIN.

13.2 PROGRAM 1 - ÇOK YÜZLÜ BİR ÇİZİMİ

Aşağıdaki program, giderek küçülen boyutta çok kenarlı bir şekil çizer.

```
10 PRINT"□"
20 CENTRE "SIMON BASIC COK YUZLU":PRINT:
PRINT
30 CENTRE "RECEP TARAFINDAN":PRINT:PRINT
40 PRINT "KENAR SAYISI ";:FETCH "□",2,N
45 IF N = 0 THEN 10
50 EXEC SPIRAL
60 TEXT10,10,"BIR TUSA BAS",1,1,8
70 GETA$:IFA$=""THEN70
80 CSET0:GOTO10
996 :
997 :
998 :
999 :
1000 PROC SPIRAL
1010 MP=100:HIRES 0,1
1020 FORJ=0TO2.5*πSTEPπ/20
1030 FORK=0+JTO2*π+J+.1STEP2*π/N
1040 X=INT(MP*1.3*SIN(K)+160)
1050 Y=INT(MP*COS(K)+100)
1060 IFK>0+JTHENLINEX1,Y1,X,Y,1
1070 X1=X:Y1=Y:NEXT
1080 MP=MP-2:NEXT
1090 END PROC
```

13.3 PROGRAM 2—KELİME ARAMA

Aşağıdaki program, kendi seçtiğiniz en fazla 20 kelimeyi girmenize olanak tanır. Ardından tüm kelimeleri bir ızgara içinde karıştırır. Göreviniz, her kelimenin başladığı koordinatları ızgara üzerinde belirlemektir. Ayrıca, oyunu bilgisayardan uzakta oynayabilmeniz için ızgarayı yazdırma seçeneğiniz de vardır.

```
10 REM*******************************
20 REM*                             *
30 REM*  KELIME ARA                 *
40 REM*                             *
50 REM*    RECEP'DEN                *
60 REM*                             *
70 REM*******************************
80 :
90 :
100 EXEC SETUP
110 EXEC GETWORDS
120 EXEC SCREEN
130 EXEC SORTLENGTHS
140 EXEC PLACEWORDS
150 EXEC PRINTGRID
160 EXEC GAME
170 CALL FINISH
180 :
181 :
3000 PROC GAME
3002 PRINT AT(28,1)"PRINT (E/H)"
3004 GETQ$:IFQ$<>"E"ANDQ$<>"H"THEN3004
3005 TI$="000000":TU=0
3006 PRINT AT(28,1)"              "
3008 IFQ$="H"THEN3010
3009 HRDCPY
3010 WF=0:REPEAT
3011 PRINT AT(28,1) "■SATIR ";:FETCH"@",
3,RO$
3012 IF TI$>"001000"THENTU=1:END PROC
3020 PRINT AT(28,1) "SUTUN ";:FETCH"@",3
,CO$
3030 PRINT AT(28,1) "              "
3040 RO=VAL(RO$):CO=VAL(CO$)
3050 IF RO>0ANDRO<21ANDCO>0ANDCO<21THEN3
070
3060 PRINT AT(28,1) "HATA":PAUSE 5000:PR
INT AT(28,1) "     ":GOTO3011
3070 F=0:FORI=1TONW:IFRO=PY(I)ANDCO=PX(I
)THENF=1:X1=I
3080 NEXT:IFF=1THEN3100
3090 PRINT AT(28,1) "YANLIS":PAUSE5000:P
RINT AT(28,1) "     ":GOTO3011
3100 FORI=0TOLEN(W$(X1))-1
3110 PRINT AT(3+PX(X1)+I*DX(TW(X1)),2+PY
(X1)+I*DY(TW(X1))) "■";
3115 PRINTMID$(W$(X1),I+1,1)
3120 NEXT:PRINT AT(25,2+X1) W$(X1):WF=WF
+1
3130 UNTIL WF=NW
```

```
3140 END PROC
3150 :
3160 :
4000 PROC PRINTGRID
4010 PRINT"▣": FORY=1TO20:FORX=1TO20
4020 IFA$(X,Y)=""THENA$(X,Y)=MID$(W$(NW*R
ND(1)+1),5*RND(1)+1,1)
4040 PRINT AT(3+X,2+Y) A$(X,Y)
4050 NEXT:NEXT
4060 END PROC
4070 :
4080 :
5000 PROC PLACEWORDS
5010 PW=0:REPEAT:PW=PW+1
5020 PX(PW)=INT(20*RND(1)+1)
5030 PY(PW)=INT(20*RND(1)+1)
5040 DR=INT(8*RND(1)+1):TW(PW)=DR
5041 CX=PX(PW)+LEN(W$(PW))*DX(DR):CY=PY(
PW)+LEN(W$(PW))*DY(DR)
5050 IFCX<1ORCX>20ORCY<1ORCY>20THEN5020
5051 REM "UYGUN.HARFLERI KONTROL EDIN."
5060 F=0:FORCK=0TOLEN(W$(PW))-1
5070 Z1$=MID$(W$(PW),CK+1,1):Z2$=A$(PX(P
W)+CK*DX(DR),PY(PW)+CK*DY(DR))
5080 IFZ2$<>""ANDZ1$<>Z2$THENF=1
5090 NEXT:IFF=1THEN5020
5091 REM ITFITS SO SLOT IT IN
5100 FORCK=0TOLEN(W$(PW))-1
5110 Z1$=MID$(W$(PW),CK+1,1):A$(PX(PW)+C
K*DX(DR),PY(PW)+CK*DY(DR))=Z1$
5120 PRINT AT(25,2+PW) W$(PW):NEXT
5130 UNTIL PW=NW
5140 END PROC
5150 :
5160 :
6000 PROC SORTLENGTHS
6010 PRINT AT(25,1) "▣SIRALAMA"
6020 F=0:FORI=1TONW-1
6030 IFLEN(W$(I))<LEN(W$(I+1))THENT$=W$(
I+1):W$(I+1)=W$(I):W$(I)=T$:F=1
6040 NEXT:IFF=1THEN6020
6050 PRINT AT(25,1) "◣          "
6060 END PROC
6070 :
6080 :
7000 PROC SCREEN
7010 PRINT"▼▼▼▼▼▼▼▼";:COLOUR 5,0
7020 PRINT"┌";:FORI=1TO20:PRINT"─";:NEXT
7030 PRINT"┐▼▌";:FORI=1TO20:PRINT"│▼▌";:
NEXT
7040 PRINT"┘▌▌";:FORI=1TO20:PRINT"─▌▌";:
NEXT
7050 PRINT"└▌▐";:FORI=1TO20:PRINT"│▌▐";:
NEXT
7060 FORI=1TO20:NU$=STR$(I):NU$=MID$(NU$
,2):IFLEN(NU$)=1THENNU$=" "+NU$
7070 PRINT AT(1,I+2) NU$:PRINT AT(I+3,0)
 LEFT$(NU$,1)
7080 PRINT AT(I+3,1) RIGHT$(NU$,1):NEXT
7090 END PROC
7100 :
```

```
7110 :
8000 PROC GETWORDS
8010 COLOUR10,0:PRINT"■■■KAÇ KELIME (MA
K 20)?■"
8020 FETCH"■",3,NW$
8030 PRINT:NW=VAL(NW$):IF NW<1ORNW>20THE
N 8010
8040 PRINT"■          SIMDI KELIMELER YAZ
IN■"
8050 PRINT" 5 ILE 15 HARF ARASINDA OLMAL
ILAR■"
8060 FORI=1TONW
8070 PRINTI"■) ";
8080 FETCH "■",15,W$(I)
8090 IFLEN(W$(I))<5THENPRINTCHR$(13)"■■"
:GOTO8070
8100 PRINT:NEXT
8110 END PROC
8998 :
8999 :
9000 PROC SETUP
9010 COLOUR 5,0:PRINT"■■AYARLAR YAPILIYO
R, LUTFEN BEKLEYIN....."
9020 DIMA$(20,20),W$(20),PX(20),PY(20),D
X(8),DY(8),TW(20)
9030 FORI=1TO20:FORJ=1TO20:A$(I,J)="":NE
XT:NEXT
9040 RESTORE:FORI=1TO8:READDX(I),DY(I):N
EXT
9050 DATA0,-1,1,-1,1,0,1,1,0,1,-1,1,-1,0
,-1,-1
9060 FORI=1TO20:W$(I)="":PX(I)=0:PY(I)=0
:NEXT
9070 END PROC
9998 :
9999 :
10000 PROC FINISH
10010 IFTU=1THEN10100
10020 PRINT AT(28,1) "TEKRAR (E/H)"
10030 GETA$:IFA$<>"E"ANDA$<>"H"THEN10030
10040 IFA$="E"THENCLR:RUN
10050 PRINT"■":END
10100 FORX1=1TONW
10110 FORI=0TOLEN(W$(X1))-1
10120 PRINT AT(3+PX(X1)+I*DX(TW(X1)),2+P
Y(X1)+I*DY(TW(X1))) "■";
10130 PRINTMID$(W$(X1),I+1,1)
10140 NEXT:PRINT AT(25,2+X1) W$(X1):NEXT
10150 GOTO10020
```

13.4 PROGRAM 3 - HARF KAYDIRICI

Bu program, 4x4'lük bir kare içinde A'dan O'ya kadar olan harfleri karıştırır. Karede bir boş alan bulunmaktadır. Harfleri kare içinde kaydırarak alfabetik sıraya göre yeniden düzenlemelisiniz.

```
1 REM ***** HARF KAYDIRMA OYUNU *****
2 REM *****                     *****
3 REM *****      RECEP'DEN      *****
4 :
5 EXEC INSTRUCTIONS
10 DIMA$(4,4),B$(4,4):MN=1
20 S$="R↑┌──Q▮▮▮  ↑  Q▮▮▮└──┘"
30 C$="■E△X↑←m"
40 L$="ABCDEFGHIJKLMNO "
50 PT=1:FORY=1TO4:FORX=1TO4
60 I1$=MID$(C$,PT,1):I2$=MID$(L$,PT,1):P
T=PT+1
70 W$=INST(I1$,S$,1)
80 W$=INST(I2$,W$,10)
90 A$(X,Y)=W$
100 NEXT:NEXT
110 A$(4,4)="■   Q▮▮▮   Q▮▮▮   "
120 PT=1:FORY=1TO4:FORX=1TO4
130 B$(X,Y)=MID$(L$,PT,1): PT=PT+1
140 NEXT:NEXT:PX=4:PY=4
150 PRINT"▽":COLOUR5,0
160 PRINT AT(12,5) "RE┌──────────────────┐";
170 FORI=1TO12:PRINT"Q▮▮";:NEXT:PRINT"Q▮
┘▮▮";
180 FORI=1TO12:PRINT"─▮▮";:NEXT:PRINT"└□
▮▮";
190 FORI=1TO12:PRINT"|□▮";:NEXT
200 EXEC LETTERS
205 EXEC SHUFFLE
206 REPEAT
210 GETR$:IFR$<"A"ORR$>"O"THEN210
215 PRINT AT(12,3) "                    "
220 EXEC CHECKIT
230 PRINT AT(12,3) MS$
240 EXEC FINISHED
250 UNTIL WI=1
260 RUN 10
998 :
999 :
1000 PROC LETTERS
1010 FORY=1TO4:FORX=1TO4
1020 PRINT AT(3*X+10,3*Y+3) A$(X,Y)
1030 NEXT:NEXT
1040 END PROC
1050 :
1060 :
2000 PROC SHUFFLE
2010 RESTORE:FORI=1TO4:READXD(I),YD(I):N
EXT
2015 PRINT AT(12,3) "EKAYDIRMA...."
2020 DATA -1,0,0,1,1,0,0,-1
2025 I=0:REPEAT:I=I+1
2030 D=INT(8*RND(RND(1))+1)
```

```
2040 IFPX+XD(D)<1ORPX+XD(D)>4ORPY+YD(D)<
1ORPY+YD(D)>4THEN2030
2050 X1=PX+XD(D):Y1=PY+YD(D):IFX2=X1ANDY
2=Y1THEN2030
2060 T$=A$(X1,Y1):A$(X1,Y1)=A$(PX,PY):A$
(PX,PY)=T$
2070 T$=B$(X1,Y1):B$(X1,Y1)=B$(PX,PY):B$
(PX,PY)=T$
2080 X2=PX:Y2=PY:PX=X1:PY=Y1
2090 EXEC LETTERS
2100 UNTIL I>100
2110 PRINT AT(12,3) "                "
2120 END PROC
2130 :
2140 :
3000 PROC CHECKIT
3005 PRINT AT(1,1) "■NUMARAYI TASI" MN
3010 FL=0:FORI=1TO4
3020 IFPX+XD(I)<1ORPX+XD(I)>4ORPY+YD(I)<
1ORPY+YD(I)>4THEN3040
3030 IFB$(PX+XD(I),PY+YD(I))=R$THENFL=1:
TE=I
3040 NEXT:IFFL=0THENMS$="KURAL DISI TASI
MA !!!":GOTO3100
3050 X1=PX+XD(TE):Y1=PY+YD(TE)
3060 T$=A$(X1,Y1):A$(X1,Y1)=A$(PX,PY):A$
(PX,PY)=T$
3070 T$=B$(X1,Y1):B$(X1,Y1)=B$(PX,PY):B$
(PX,PY)=T$
3080 EXEC LETTERS
3090 MS$="OLDU":PX=X1:PY=Y1:MN=MN+1
3100 END PROC
3110 :
3120 :
4000 PROC FINISHED
4010 FI$="":FORY=1TO4:FORX=1TO4
4020 FI$=FI$+B$(X,Y):NEXT:NEXT
4030 IFFI$=L$THEN4050
4040 END PROC
4050 PRINT AT(12,3) "BIR KAZANAN !!!":PA
USE10:WI=1:GOTO4040
4060 :
4070 :
5000 PROC INSTRUCTIONS
5005 COLOUR10,0
5010 PRINT"■▲TALIMATA IHTIYAC VAR MI (E/
H)?"
5020 GETA$:IFA$<>"E"ANDA$<>"H"THEN5020
5030 IFA$="N"THEN END PROC
5040 PRINT"■■■ OYUNUN AMACI "
5050 PRINT"■TUM HARFLERI "
5060 PRINT"■DOGRU SIRAYA KOYMAKTIR...."
5070 PRINT"■■                    A B C D "
5080 PRINT"■                    E F G H "
5090 PRINT"■                    I J K L "
5100 PRINT"■                    M N O ※"
5110 PRINT"■■■   BOS ALANA BIR HARF"
5120 PRINT"■   TASIMAK ICIN, O HARFI"
5130 PRINT"■   KLAVYEDE YAZMANIZ YETER."
5140 PRINT"■■   SIMDI DENEYIN."
5150 GETA$:IFA$=""THEN5150
5160 END PROC
```

13.4 PROGRAM 4 - KLASİK BİR ARABA

Bu program, çok renkli grafik ekranında klasik bir araba çizer.

```
1 COLOUR10,5
10 HIRES0,1:MULTI0,2,1
20 CIRCLE30,150,10,11,1
30 CIRCLE30,150,13,15,1
31 PAINT28,150,2
32 PAINT19,150,1
35 ARC30,150,270,90,10,17,21,3
36 LINE15,150,17,150,3
37 LINE43,150,45,150,3
38 PAINT16,148,3
40 BLOCK3,105,13,150,1
41 LINE3,105,6,105,0
42 LINE3,105,3,107,0:LINE 10,105,13,105,
0:LINE13,105,13,107,0
43 LINE3,150,6,150,0
44 LINE3,148,3,150,0
45 LINE3,115,0,125,1
46 LINE3,140,0,125,1
50 LINE16,139,16,105,1
60 LINE16,105,30,105,1
70 LINE30,105,30,129,1
71 PAINT18,107,1
80 LINE30,105,70,105,2
90 LINE70,105,70,145,2
91 LINE70,145,47,145,2
92 PAINT33,107,2
93 LINE43,152,56,152,1
94 BLOCK56,152,74,154,1
95 LINE74,152,80,152,1
96 LINE80,152,88,144,1
98 BLOCK72,105,94,145,2:PLOT71,105,1
99 PAINT75,151,1
100 BLOCK82,146,108,153,1
101 BLOCK110,144,120,152,1
102 BLOCK95,105,108,146,1
103 BLOCK108,105,114,141,1
105 BLOCK31,70,36,104,1
106 LINE31,70,95,70,1
```

```
107 BLOCK70,70,72,104,2
108 BLOCK31,67,95,69,2
109 LINE69,71,69,104,1
110 LINE72,71,72,104,1
112 LINE95,70,95,104,1
114 BLOCK35,102,69,i04,1
115 BLOCK73,102,95,104,1
116 LINE88,102,91,96,1
120 LINE31,66,76,66,1
121 LINE31,65,46,65,1
122 LINE31,64,36,64,1
132 LINE97,70,99,73,1
133 LINE97,67,99,73,1
134 LINE97,68,99,73,1
135 LINE97,69,99,73,1
220 CIRCLE140,150,10,11,1
230 CIRCLE140,150,13,15,1
231 PAINT138,150,2
232 PAINT129,150,1
235 ARC140,150,270,90,10,17,21,3
236 LINE125,150,127,150,3
237 LINE153,150,155,150,3
238 PAINT126,148,3
300 LINE128,152,114,152,1
310 LINE114,144,124,144,1
320 PAINT127,151,1
330 LINE125,141,116,141,1
340 LINE116,141,116,115,1
341 LINE116,115,136,115,1
342 LINE136,115,136,130,1
345 PAINT133,127,1
350 BLOCK116,105,136,112,1
360 LINE137,105,141,105,2
370 LINE141,105,141,129,2
380 LINE137,105,137,115,2
390 PAINT139,107,2
391 PLOT141,105,1:FORT=128TO120STEP-2
392 LINET,118,T,127,3:NEXT
400 LOW COL0,2,6+8:FORT=0TO159:IFTEST(T,
98)=0THENPLOTT,98,3
410 NEXT
420 FORT=0TO62STEP8:LINE0,T,160,T,3:NEXT
:BLOCK0,0,160,62,3
430 PAINT2,64,3
500 FORT=0TO159STEP5:IFTEST(T,96)=0THENP
AINTT,96,3
510 NEXT
600 LINE78,107,76,107,1:PLOT76,108, 1
610 LOW COL7,2,6+8
620 ARC0,0,90,180,10,20,20,1
630 PAINT0,0,1
700 FORT=90TO180STEP10:ANGL0,0,T,33,33,1
:NEXT
999 PAUSE4:NRM:MEM:COLOUR0,0:PRINT"[illegible]
[illegible]GOR..!!"
```

EK A

HATA MESAJLARI

SIMON BASIC'i kullanırken bir hata mesajı görünebilir. Bu mesajlar SIMON BASIC kartuşuna özgüdür. Her hata mesajı, anlamı ve olası nedeni bu ekte verilmiştir.

? BAD MODE

Bu durum, bir komuttaki herhangi bir parametrenin izin verilen aralığın dışında olması halinde ortaya çıkar.

? NOT HEX CHARACTER

Onaltılık olmayan bir sayıyı ondalık karşılığına dönüştürme girişiminde bulunulmuştur.

? NOT BINARY CHARACTER

İkili olmayan bir sayıyı ondalık karşılığına dönüştürme girişiminde bulunuldu.

? UNTIL WITHOUT REPEAT

UNTIL komutu, önceden tanımlanmış bir REPEAT komutu olmadan kullanılmıştır.

? END LOOP WITHOUT LOOP

END LOOP komutu, önceden tanımlanmış bir LOOP döngüsü olmadan kullanılmıştır.

? END PROC WITHOUT EXEC

END PROC komutu, herhangi bir işlem yürütülmeden kullanılmıştır.

? PROC NOT FOUND

Var olmayan bir yöntemi seçme girişiminde bulunulmuştur.

? NOT ENOUGH LINES

MOB tasarım ızgarası için yeterli sayıda çizgi oluşturulmadı.

? BAD CHAR FOR A MOB IN LINE n

MOB tasarım aşamasındaki bir parametre tanımlanan aralığın dışında. Hatanın satır numarası her zaman DESIGN komutunun yürütüldüğü satır numarasıdır, ancak bu mutlaka hatanın o satırda olduğu anlamına gelmez.

? STACK TOO LARGE

Bu durum, dokuzdan fazla iç içe prosedür veya program döngüsü kullandığınızda ortaya çıkar.

SÖZLÜK

Bu kılavuzda kullanılan terimlerin listesi.

BIT
'İkili Basamak'ın kısaltması. Bilgisayar belleğinin en küçük birimi.

BRANCH
Program yürütmesinin bir satırdan diğerine aktarılması.

COORDINATE
Bir koordinat sistemindeki bir noktanın x veya y eksenine olan uzaklığı.

DATA
Bilgisayarın belleğinde veya bir depolama aygıtında tutulan bilgiler.

DEBUGGING
Programlama hatalarını düzeltme.

FLOATING POINT
Sayıları üslü biçimde tutan bir sistem.

INTEGER
Tam sayı.

KERNAL
COMMODORE 64'ün işletim sistemi.

LIBRARY
Programlar ve/veya program alt rutinlerinden oluşan bir stok.

MOB
Hareket ettirilebilir nesne bloğu. Bir yaratığı tanımlamak için kullanılan bir terim.

OCTAVE
Sekiz müzik notasından oluşan bir grup.

ORIGIN
Ekranda bir şeklin çizildiği nokta.

PERIPHERAL
Bilgisayara bağlanan harici bir cihaz.

PIXEL
Ekrandaki en küçük adreslenebilir konum.

PROGRAM CRASH
Program yürütmesinde istenmeyen bir duraklama.

REGISTER
Bilgisayarın belleğinde ayrılmış bir alan.

SECONDARY ADDRESS
Bir program/dosya depolama talimatı.

SCROLLING
Ekran üzerinde dikey veya yatay yönde hareket etme.

SPRITE
Programlanabilir bir nesne.

START ADDRESS
Bir veri bloğunun bilgisayarın belleğinde depolanmaya başlandığı nokta.

TIME CYCLE
Bir frekans bileşeninin süresi, saniyenin binde biri cinsinden ölçülür.

İNDEKS

ANGL 6-1,6-12
ARC 6-1,6-11
AT 3-1,3-6
AUTO 2-1,2-3

Baskı
- grafik ekranındaki karakterler 6-1,6-18
- grafik ekranındaki karakter dizileri 6-1,6-19
- ekran verileri 7-1,7-11

BCKGNDS 7-1,7-2
Bestelenmiş müzik çalmak 11-11
BFLASH 7-1,7-4
BLOCK 6-1,6-14
Block, Data 8-2

CALL 9-1,9-5,9-6
CENTRE 3-1,3-5
CGOTO 2-1,2-6
CHAR 6-1,6-18
CHECK 8-1,8-9
CIRCLE 6-1,6-10
CMOB 8-1,8-5
COLD 2-1,2-15
COLOUR 6-1,6-3
COPY 7-1,7-11
CSET 6-1,6-17

Çarpışma tespiti, MOB 8-1,8-8,8-9
Çizim
- bir yay 6-1,6-11
- dairesel şekiller 6-1,6-10
- bir dairenin yarıçapı 6-1,6-12
- tek noktalar 6-1,6-8

Çizim renklerini değiştirme 6-1,6-6
Çizim renkleri 6-1,6-2
Çizim türleri 6-3
Çok renkli grafikler 6-1,6-5
Çokyüzlü Çizimi 13-1

Dalga formu
- Darbe/Kare 11-3
- Gürültü 11-4
- Testere dişi 11-3
- Üçgen 11-2

Darbe/Kare dalga formu 11-3

Değişken frekans 11-6
DELAY 2-1,2-10
DESIGN 8-1,8-2,8-10,8-12
DETECT 8-1,8-8
DIR 5-2
DISABLE 3-1,3-11
DISAPA 2-1,2-16
DISK 5-1
DISPLAY 2-1,2-3
DIV 4-1,4-2
Disket biçimlendirme 5-1
Disketten SIMON BASIC yükleniyor 1-4
Disket dizini
    tüm 5-1,5-2
    secilen 5-1,5-2
Dizi olmayan değişkenlerin görüntülenmesi 2-1,2-14
Dosyayı karalamak 5-2
DRAW 6-1,6-14
Dikdörtgen çizmek 6-1,6-5
Dönüştürmeler 1-2,1-6
DUMP 2-1,2-14
Durum testi 9-1
DUP 3-1,3-5

Ekranın bir alanını kaydırma 7-1,7-9
Ekranın bir bölümünü çoğaltma 7-1,7-7
Ekran kenar renginin yanıp sönmesi 7-1,7-4
Ekran renginin yanıp sönmesi 7-1,7-3
Ekran verilerini saklama 7-1,7-10
Ekran verilerini tersine çevirme 7-1,7-8
END PROC 9-6
ENVELOPE 11-1,11-4,11-8
Etiketleme programı rutinleri 9-1
EXEC 9-1,9-5,9-7

FCHR 7-1,7-5
FCOL 7-1,7-6
FILL 7-1,7-6
FLASH 7-1,7-3
Fonksiyon tuşlarına komut atama 2-2
Fonksiyon tuşunun test edilmesi 3-9
FRAC 4-1,4-2

GLOBAL 9-8

Harf Kaydırıcı programı 13-5
Hata yakalama 10-1
Hatırlamak
    saklanan ekran verileri 7-1,7-10,7-11
    yeni bir program 2-1,2-18
    Son grafik ekranının yeniden görüntülenmesi 6-1,6-17

HI COL 6-1,6-7
HIRES 6-1,6-4
HRDCPY 7-1,7-12

IF...THEN...ELSE 9-1
İkili sayıdan ondalık sayıya dönüştürme 4-1,4-3
INKEY 3-1,3-9
INSERT 3-1,3-2
INST 3-1,3-3
INV 7-1,7-8
Işık kalem 12-1

İşlev tuşu komutlarının listelenmesi 2-1,2-3

JOY 12-1,12-5
Joystick 12-1,12-5

Kalabalığı hareket ettirmek 8-1,8-7,8-8
Kapalı bir alanı renkle doldurmak 6-1,6-13
Karakter dizilerini çoğaltma 3-1,3-5
Karakter kümesi seçimi 6-1,6-17
Karakter tasarım ızgarası oluşturma 8-12
Karakterin rengini değiştirmek 7-1,7-6
Kelime arama programı 13-2
KEY 2-1,2-2
Klasik otomobil programı 13-8
Koordinatlar 6-2
Kullanıcı tanımlı karakterler 8-1,8-10
Kürekler 12-1,12-3
Küresel değişkenler 9-8

LINE 6-10
LOCAL 9-8
LOOP...EXIT IF...END LOOP 9-1,9-4
LOW COL 6-1,6-6

MEM 8-10
MERGE 2-1,2-7
Metni ortalama 3-1,3-5
MMOB 8-1,8-7
MOB'un başlatılması 8-6
MOB çarpışma tespiti 8-1,8-8,8-9
MOB OFF 8-9
MOB önceliği 8-6
MOB SET 8-1,8-6
MOB tasarım ızgarası oluşturma 8-2
MOB'u temizlemek 8-9
MOD 4-1,4-2

MOVE 7-1,7-7
MULTI 6-1,6-5
MUSIC 11-4,11-9
Müzik ses seviyesini seçme 11-4,11-5
Müzik
daireler 11-9
dinlenmeler 11-9
keskinler 11-9

NO ERROR 10-1,10-4
NRM 6-6

OFF 7-1,7-4
Okuma fonksiyonları 12-1
OLD 2-1,2-18
Onaltılık sayıdan ondalık sayıya dönüştürme 4-1,4-3
ON ERROR 10-1
ON KEY 3-1,3-10
OPTION 2-1,2-9
Otomatik program satırı numaralandırması 2-1,2-3
OUT 10-1,10-3

PAGE 2-1,2-8
PAINT 6-1,6-12
PAUSE 2-1,2-5
PENX 12-1
PENY 12-1,12-2
PLACE 3-1,3-4
PLAY 11-4,11-11
PLOT 6-1,6-8
POT 12-1,12-3
PROC 9-1,9-5
Prosedürler 9-5
Program döngüleri 9-1
Program kodunu gizleme 2-15,2-16
Programlarda hata ayıklama 2-12
Fonksiyon tuşlarının programlanması 2-2
Programlama sesi 11-4

RCOMP 9-1,9-3
REC 6-1,6-5
Renk, Çizim 6-1,6-3
RENUMBER 2-1,2-4
REPEAT UNTIL 9-1,9-2

RESET 2-1,2-6
RESUME 3-1,3-11
RETRACE 2-1,2-13
Halka Modülasyonu 11-7
RLOCMOB 8-1,8-8
ROT 6-1,6-4

Sayısal veriler, Biçimlendirme 3-1,3-7
SCRLD 7-1,7-10,7-11
SCRSV 7-1,7-10
SECURE 2-1,2-15,2-17
Sesin 'şeklini' tanımlamak 11-1
SIMON BASIC kartuşunu takma 1-4
Statik frekans 11-6
Senkronizasyon 11-6

Şekil tasarlamak 6-1,6-14

Tamamen gölgelendirilmiş bir renk bloğu çizmek 6-1,6-14
Tam sayı bölme 4-1
TEST 6-1,6-9
Test bit 11-7
Testere dişi dalga formu 11-3
TEXT 6-1,6-19
TRACE 2-1,2-12

USE 3-1,3-7

Üçgen dalga formu 11-2,11-7

Veri bloğu 8-2
VOL 11-4,11-5

WAVE 11-4,11-5

Yerel değişkenler 9-8
Yüksek çözünürlüklü grafikler 6-1,6-4

Zarf Oluşturucu 11-6

commodore
COMPUTER